Reitinhaku maastokartoilla

Santeri Karjalainen

Tampereen yliopisto
Luonnontieteiden tiedekunta
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Martti Juhola
Toukokuu 2017

Tampereen yliopisto
Luonnontieteiden tiedekunta
Tietojenkäsittelytieteiden tutkinto-ohjelma
Reitinhaku maastokartoilla
Santeri Karjalainen
Pro gradu -tutkielma, 88 sivua
Toukokuu 2017

Reitinhakualgoritmien avulla voidaan etsiä optimaalisia polkuja monenlaisissa hakuavaruuksissa. Erilaiset kartat ovat tyypillisiä sovelluskohteita reitinhakualgoritmeille. Tässä tutkielmassa pyrin selvittämään, miten maastoa kuvaavaan paikkatietoaineistoon voidaan kohdistaa reitinhakualgoritmeja optimaalisen reitin löytämiseksi. Käsittelen myös korkeusmallien huomioimisen optimaalisten reittien haussa.

Paikkatietoaineisto tarvitsee reitinhakua varten muuttaa reitinhakualgoritmille soveltuvaan muotoon. Keskityn aineiston käsittelyn suhteen erityisesti rasterointimenetelmään, johon liittyen esittelen erilaisia ruudukointitapoja ja soveltuvia reitinhakualgoritmeja. Käyn läpi myös rasterointimenetelmän kaltaiset navigointiverkot ja kulmariippumattoman reitinhaun.

Esittelen toteuttamani sovelluksen, joka käyttää kuvailemiani menetelmiä reitinhakuun Maanmittauslaitoksen maastotietokannasta. Sovelluksen suorituskyky noudattaa rasterointimenetelmästä johdettavissa olevaa kompleksisuutta. Esimerkiksi $50\,\mathrm{m} \times 50\,\mathrm{m}$ ruudukolla 37,9 kilometriä pitkän reitin laskeminen kestää noin kahdeksan sekuntia, aineiston esiprosessoinnin kestäessä muutamia minuutteja. Sovelluksen tuottamat reitit ovat käytettyjen parametrien ja karttatasojen suhteen optimaalisia, mutta vasta todellisia eri maastonkohtien liikkumiskustannuksia mittaamalla voitaisiin saavuttaa esimerkiksi metabolisten kustannusten suhteen optimaalisia reittejä.

Sisällys

1 Johdanto

Reitinhaku on ollut automaattiselle tietojenkäsittelylle merkittävä sovellusalue 1950-luvulta asti. Reitinhakua tarvitaan lukuisissa eri sovelluksissa, ja reitinhaku on olennainen osa myös erilaisia kuluttajakäyttöön tarkoitettuja järjestelmiä. Autonomisesti liikkuvien kulkuneuvojen, kuten autojen, tarvitsee soveltaa reitinhakualgoritmeja, jotta ne voivat suunnitella korkealla tasolla optimaalisia reittejä. Lisäksi monissa eri tuotantoteollisuuden prosesseissa tarvitsee esimerkiksi optimoida erilaisten kuljettimien reittejä. Rakentamisessa taasen reitinhakua voidaan käyttää esimerkiksi jonkin tieosuuden rakentamiskustannusten minimoimiseksi.

Erilaisissa hakuavaruuksissa toimivia reitinhakualgoritmeja on useita, ja ne käsittelevät pääasiassa graafeja. Automaattinen reitinhaku ei välttämättä optimoi reitin pituutta, vaan optimoitava ominaisuus voi olla mikä tahansa. Eri ominaisuuksilla käytetään erilaisia menetelmiä, mutta reitinhaun ongelmaan liittyy joitakin yleisiä periaatteita sovellusalueesta riippumatta.

Maastokarttojen reitinhaun haasteellisuus johtuu ensisijaisesti siitä, että hakualgoritmin hakuavaruus ei ole diskreetti. Tieverkostolla navigoitaessa hakuavaruus voidaan diskretisoida risteysten suhteen, mutta maastokartoilla näin voitaisiin toimia ainoastaan polkujen osalta. Siinä missä tietä pitkin voidaan liikkua vain kahteen suuntaan, maastokartan peittämällä alueella suuntia on äärettömästi. Lisäksi tieverkostolla reitit muodostuvat aina risteysten välille. Maastokartoilla sen sijaan voidaan edetä mielivaltaisesti minkä tahansa kahden pisteen välillä. Hakuavaruus on siis valtava, ja sen diskretisointi on keskeisimpiä haasteita riittävän täsmällisen ja nopean reitinhaun mahdollistamiseksi.

Jotta reitinhakualgoritmeja voidaan soveltaa johonkin karttaan, tulee karttaaineisto muuttaa reitinhakualgoritmille soveltuvaan muotoon. Kartat koostuvat paikkatiedosta, jonka käsittelyyn tarvitaan paikkatietojärjestelmiä. Käsittelen paikkatietojärjestelmiä luvussa 2, jossa esittelen myös muutamia avoimen- ja suljetun lähdekoodin paikkatietosovelluksia. Paikkatiedon ominaisuuksia ja tallentamista käsittelen luvussa 3, jossa esittelen myös korkeusmallit. Luvussa 4 käsittelen seikkoja, joita tulee huomioida erilaisia karttoja käytettäessä.

Keskityn reitinhaun ja paikkatiedon käsittelyn osalta erityisesti rasterointimenetelmään, mutta vertailen myös vektori- ja rasterimenetelmien eroavaisuuksia. Luvussa 5 esittelen muutamia rasterointimenetelmään soveltuvia reitinhakualgoritmeja ja ruudukointikeinoja. Käsittelen lisäksi kulmariippumatonta reitinhakua ja videopeleissä käytettäviä navigointiverkkoja, joita voidaan soveltaa tarvittaessa mihin

tahansa kartta-aineistoon.

Luvussa 6 käsittelen aiempaa maastokarttojen reitinhakuun kohdistunutta tutkimusta. Tieverkostot ovat reitinhaulle huomattavasti yleisempi sovelluskohde niiden diskreetin luonteen takia, mutta myös maastokarttoja on käytetty aineistona optimaalisten reittien haussa. Lopuksi luvussa 7 esittelen toteuttamani sovelluksen, joka hyödyntää esittelemiäni menetelmiä ja algoritmeja reitinhaussa Maanmittauslaitoksen maastotietokannasta.

2 Paikkatietojärjestelmät

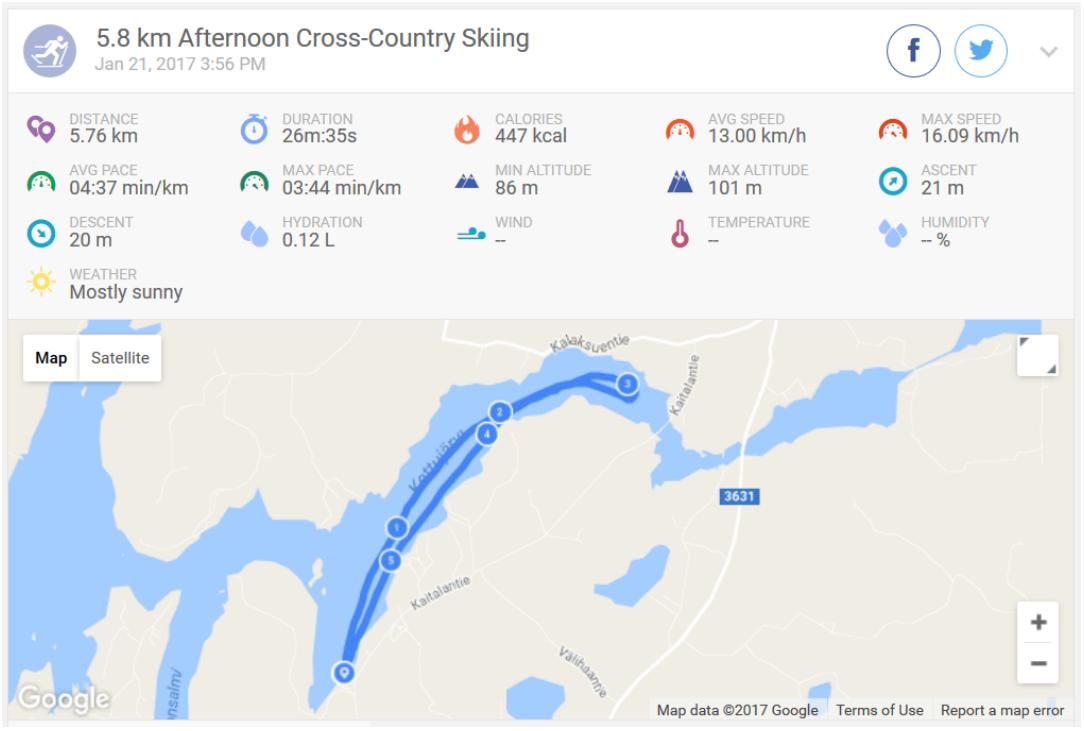
Paikkatietojärjestelmät (*geographic information system, GIS*) ovat tietojärjestelmiä, jotka mahdollistavat paikkatiedon (*geographic information* tai *geospatial information*) hyödyntämisen erilaisissa käyttötarkoituksissa. Paikkatiedolla tarkoitetaan tietoa maapallon pinnalla olevien kohteiden sijainneista. Paikkatietojärjestelmät, kuten tietojärjestelmät yleisestikin, ovat laajoja kokonaisuuksia jotka koostuvat automaattisen tietojenkäsittelyn mahdollistavan laitteiston ja ohjelmiston lisäksi myös ihmisistä ja käytännöistä. Paikkatietoa ja paikkatietojärjestelmiä tutkivaa tieteenalaa kutsutaan geoinformatiikaksi (*geoinformatics*). (Sanastokeskus TSK, 2014; Heywood, Cornelius, & Carver, 2006, s.18–21)

Tässä luvussa esittelen paikkatietojärjestelmän keskeisimmät osat keskittyen erityisesti paikkatietosovelluksiin ja niiden toimintaan sekä ominaisuuksiin. Reitinhaussa maastokartoilla keskeisimpinä osina voidaan pitää reitinhaun toteuttavaa sovellusta, valittuja algoritmeja sekä paikkatietoaineistoa (*spatial dataset*). Koska algoritmilla ja sovelluksella on aina jokin käyttötarkoitus ja kohderyhmä, myös käyttäjät tulee huomioida. Reitinhakua maastokartoilla voidaan siis kuvailla myös paikkatietojärjestelmänä, vaikkakin keskityn tutkielmassani erityisesti aiheen tietojenkäsittelylliseen näkökulmaan.

2.1. Paikkatietojärjestelmien historiaa

Paikkatietojärjestelmiä käytetään nykyään laajalti eri tieteenaloilla sekä teollisuudessa. Ne siirtyivät pikkuhiljaa yliopistoista ja tutkimuslaitoksista ensin sotateollisuuden ja viranomaisten käyttöön 1980-luvulla, ja edelleen kaupalliseen käyttöön 1990-luvulla. Paikkatietojärjestelmien käyttöönoton historiaan voidaan soveltaa Rogersin (2003) tunnettua innovaatioiden diffuusioteoriaa. (Longley, Goodchild, Maguire, & Rhind, 2005, s. 41)

Kuluttajakäyttöön tarkoitetut paikkatietojärjestelmät ja -sovellukset ovat yleistyneet nopeasti 2000-luvulla mobiililaiteteknologian kehittymisen myötävaikutuksesta. Kuvassa 1 on esitetty kuvakaappaus mobiilisovelluksella tallennetusta liikuntasuorituksesta. Erilaiset liikuntasuorituksien seurantaan käytetyt sovellukset ovat saavuttaneet merkittävän aseman osana ihmisten jokapäiväistä elämää. Niissä keskeisenä elementtinä on paikkatiedon kerääminen ja edelleen liikuntasuoritukseen liittyvän tiedon johtaminen paikkatiedosta. Käyttäjämäärät ovat suuria; Under Armourin ostaessa MyFitnessPal- ja Endomondo-sovellukset niillä oli yhteensä noin



Kuva 1. Liikuntasuoritusten seurantaan suunniteltu paikkatietojärjestelmä, Endomondo. (Endomondo, 2017)

120 miljoonaa käyttäjää. (Under Armour, 2015)

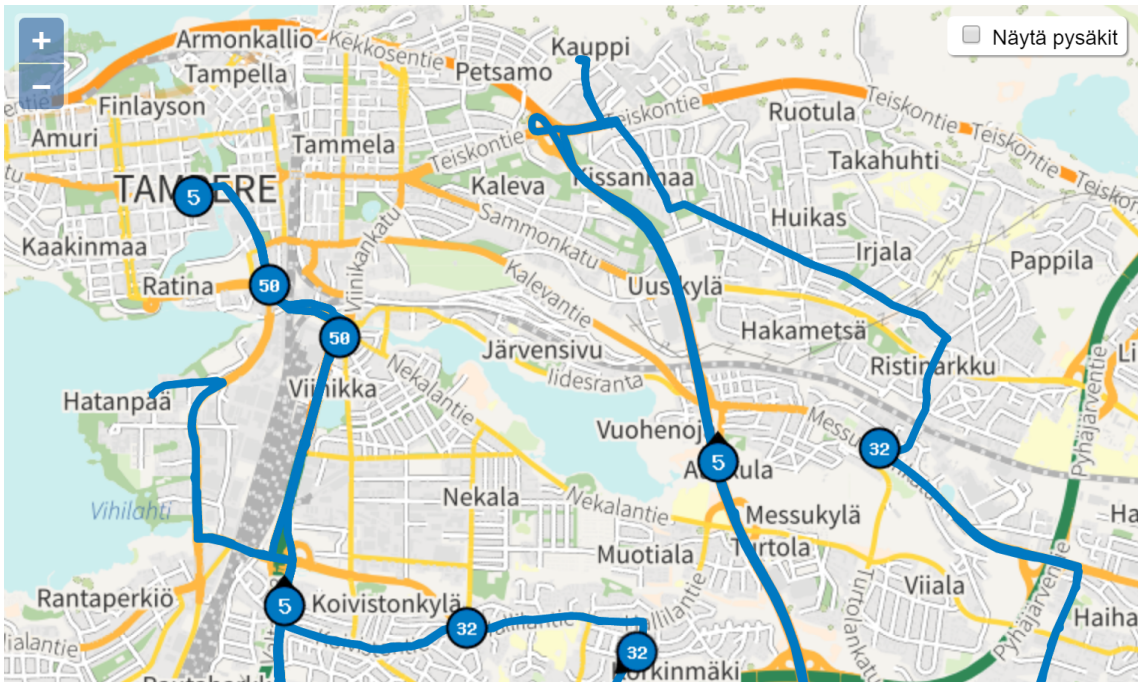
2.2. Paikkatietojärjestelmien luokittelu

Sanastokeskus TSK:n Geoinformatiikan sanaston (2014) mukaan paikkatietojärjestelmät voidaan jakaa kahteen kategoriaan niiden käyttötarkoituksen mukaan: ”tapahtuma- ja tiedonhallintapainotteisiin tietojärjestelmiin” ja ”analyysipainotteisiin päätöksenteon tukijärjestelmiin”. Osa paikkatietojärjestelmistä saattaa luonnollisesti tarjota työkaluja molempiin käyttötarkoituksiin. Esimerkiksi kuvassa 1 esitetyn Endomondo-sovelluksen voidaan ajatella kuuluvan molempiin kategorioihin. Päätöksenteon tuki ilmenee siten, että sovellus tarjoaa tukea käyttäjän harjoittelun suunnitteluun paikkatiedosta johdetun suorituskykytiedon perusteella. Toisaalta sovellus kerää käyttäjän liikuntasuorituksiin liittyvää paikkatietoa, joten sen voidaan katsoa kuuluvan myös tapahtuma- ja tiedonhallintapainotteisiin tietojärjestelmiin.

Esimerkkinä tapahtuma- ja tiedonhallintapainotteisesta paikkatietojärjestelmästä voidaan mainita vaikkapa erilaiset ajoneuvojen seurantaan käytetyt järjestelmät, jotka perustuvat usein GPS-satelliittipaikannukseen. Tällaisia järjestelmiä tarjoaa esimerkiksi Navicom Oy, joka on suomalainen ajoneuvoseurantaan erikoistunut yri-

tys. Navicomin tarjoama paikkatietojärjestelmä ylläpitää muun muassa ajoneuvojen reaaliaikaista paikkatietoa sekä paikkatietohistoriaa. (Navicom, 2017)

Reitinhaku maastokartoilla kuuluu selkeästi analyysipainotteisen päätöksenteon tukijärjestelmien piiriin. Reitinhaku on kuluttajakäytössä yleinen paikkatietojärjestelmien sovellusalue. Tieliikenteessä käytetään runsaasti erilaisia auto- ja moottoripyöränavigaattoreita. Tiellä tai maastossa navigoitaessa nämä paikkatietojärjestelmät tarjoavat toiminnallisuuden paikkatietoaineiston analysoimiseksi ja tukevat käyttäjän päätöksentekoa reittivalintojen suhteen. Reitinhakua toteuttaviin päätöksenteon tukijärjestelmiin kuuluvat myös erilaiset julkisen liikenteen reittejä laskevat paikkatietosovellukset, kuten esimerkiksi Tampereen seudun joukkoliikenteen reittiopas. (Tampereen seudun joukkoliikenne, 2017b)



Kuva 2. Linja-autojen seurantaan tarkoitettu Lissu Liikenteenseuranta. (Tampereen seudun joukkoliikenne, 2017a)

Toisaalta esimerkiksi Goodchild (2009b) jaottelee paikkatietojärjestelmät neljään eri kategoriaan. Näistä ensimmäinen on paikannus (*positioning*), joka koskee erilaisten kohteiden paikannusta esimerkiksi satelliittien tai radiotaajuudella toimivien etätunnistimien avulla. Erilaisten paikannusmenetelmien avulla saadaan tärkeää tietoa maapallolla liikkuvista kohteista. Tämä mahdollistaa esimerkiksi kuvassa 2 esitetyn kaltaiset seurantapalvelut, joista asiakkaat voivat seurata reaaliaikaisesti linja-autojen sijaintia.

Toinen kategoria on tiedonkeruu. Tähän kategoriaan kuuluvat erityisesti ortoilmakuvaus ja satelliittien käyttämät maapallon pintaa tarkkailevat kuvausvälineet. Myös esimerkiksi laseria käyttäviä kaukokartoituslaitteita voidaan käyttää maapallon pinnan äärimmäisen tarkkaan kuvaamiseen. Äärimmäisen tarkoilla sähkömagneettista spektriä havainnoivilla antureilla voidaan kerätä paljon erilaisia maapallon pintaa kuvaavia muuttujia. (Goodchild, 2009b)

Kolmas Goodchildin (2009b) paikkatietojärjestelmäkategoria on tiedonlevitys. Paikkatietoa on perinteisesti levitetty erilaisten karttojen muodossa. Paperikartat ovat edelleen laajalti käytössä, mutta niiden rinnalle ovat tulleet digitaaliset kartat. Nykyisin paikkatietoa on merkittävissä määrin saatavilla avoimena datana. Erilaiset paikkatietoportaalit kuten Maanmittauslaitoksen (2017b) Karttapaikka-palvelu toimivat avoimen datan julkaisualustoina. Neljäs Goodchildin kategoria on analyysi, joka kattaa kaikki paikkatiedon analysointiin kykenevät järjestelmät. Näitä ovat esimerkiksi reitinhakujärjestelmät ja maan pinnan eroosiota tutkivat järjestelmät.

2.3. Paikkatietosovelluksia

Työpöytäkäyttöön on saatavilla sekä avoimen että suljetun lähdekoodin monipuolisia paikkatietosovelluksia. Avoimen lähdekoodin paikkatietosovelluksien määrä ja markkina-asema on kasvanut merkittävästi viime vuosina. Tämä näkyy esimerkiksi valtiollisten tahojen rahoittamien avoimen lähdekoodin projektien muodossa. Avointen paikkatietojärjestelmien käyttäjämäärät ovat myös kasvaneet merkittävästi. Yhä suurempi osa paikkatietojärjestelmiin kohdistuvasta tutkimuksesta julkaisee tuloksensa avoimella lähdekoodilla, joka osaltaan edesauttaa järjestelmien kehitystä. (Steiniger & Hunter, 2013)

Tässä kohdassa esittelen muutamia vapaita ja avoimen lähdekoodin paikkatietosovelluksia sekä kaupallisia ja suljettuja paikkatietosovelluksia. Kaikilla tässä luetelluilla sovelluksilla on mahdollista laskea lyhyimpiä reittejä. Reitin laskenta vaatii kuitenkin paikkatietoaineiston esikäsittelyä reitinhakualgoritmeille sopivaksi. Erilaisten tie- ja polkuverkkojen reitit ovat yleinen käyttökohde näille sovelluksille ja tällaiseen käyttötarkoitukseen ne soveltuvat hyvin. Maastokartoilla reitinhaku on kuitenkin huomattavasti hankalampaa, joskin mahdollista näillä sovelluksilla.

2.3.1. Avoin lähdekoodi

Avoimen lähdekoodin paikkatietosovelluksien ja -järjestelmien parissa toimivista kattojärjestöistä merkittävimpänä voidaan mainita OSGeo (*The Open Source Geos-*

patial Foundation). Sen tavoitteena on tukea avoimen lähdekoodin paikkatietojärjestelmien yhteisöllistä kehitystä ja kannattaa tällaisten järjestelmien laajamittaista käyttöä. Kyseessä on voittoa tavoittelematon järjestö, joka yhdistää useita paikkatietojärjestelmäprojekteja erilaisten standardien ja toimintamenetelmien avulla. (The Open Source Geospatial Foundation, 2017b)

OSGeo järjestää vuosittain FOSS4G-konferenssin, jossa käsitellään vapaiden ja avoimien paikkatietojärjestelmien kehitystä. Konferenssiin osallistuu ohjelmistokehittäjien lisäksi käyttäjiä ja muita avainhenkilöitä eri organisaatioista ja monilta eri sovellusalueilta. Konferenssin tavoitteena on luoda tehokkaita paikkatietotuotteita, standardeja ja protokollia. Vuoden 2017 konferenssi järjestetään Bostonissa, Yhdysvalloissa. (The Open Source Geospatial Foundation, 2017a)

Toinen merkittävä avoimen lähdekoodin paikkatietojärjestelmiin liittyvä organisaatio on OGC (*Open Geospatial Consortium*). OGC on voittoa tavoittelematon järjestö, joka pyrkii tuomaan yhteen paikkatietojärjestelmiä käyttäviä tieteen- ja teollisuudenaloja. OGC kehittää erilaisten pilottiprojektien ja tutkimuksen avulla standardeja, jotka mahdollistavat esimerkiksi erilaisten paikkatietojärjestelmien yhteensopivuuden. OGC on kehittänyt muun muassa GML:n (*Geography Markup Language*), jota käytetään laajalti maantieteen kuvaamiseen, tallentamiseen ja siirtämiseen. Kaikki OGC:n määrittelemät standardit ja niihin liittyvät tieteelliset julkaisut ovat helposti saatavilla internetistä. Määrittelyihin kuuluu myös useita ohjelmointirajapintoja, joita eri paikkatietojärjestelmät toteuttavat. (Open Geospatial Consortium, 2017b)

GRASS GIS

GRASS GIS (*Geographic Resources Analysis Support System*) on yksi merkittävimmistä OSGeon projekteista. Sitä on kehitetty vuodesta 1982 asti. Kehitykseen ovat oman panoksensa antaneet useat Yhdysvaltojen liittovaltion virastot sekä yliopistot ja yksityiset yritykset. GRASS GIS:n kehitys siirtyi 90-luvun lopulla akateemisten instanssien vastuulle. (GRASS development team, 2017) Steinigerin ja Hunterin (2013) mukaan GRASS GIS:llä on paljon käyttökohteita, mutta sen käytettävyys on rivikäyttäjän näkökulmasta heikko. Tuki Windows-alustalle esiteltiin vasta versiossa 6.0.3. Sovellus sisältää kuitenkin mittavan määrän paikkatietoaineistojen käsittelyyn tarvittavia toimintoja. Se kilpaileekin toiminnallisuudeltaan esimerkiksi kaupallisen ArcGIS-järjestelmän edistyneimmän ArcGIS Desktopin kanssa. GRASS GIS:n käyttäjäkunta muodostuu pääosin tutkimuslaitoksista ja yliopistoista.

GRASS GIS tarjoaa toimintoja muun muassa rasteri- ja vektorianalyysiin, ilmakuvien prosessointiin, koneoppimiseen, geokoodaukseen, visualisointiin ja geostatistiikkaan. Kolmiulotteiselle datalle löytyy tuki esimerkiksi optisten kaukokartoituslaitteiden pisteparvien ja kolmiulotteisten rasterien analyysin muodossa. Myös lukuisille tietokantateknologioille sopivat rajapinnat ovat osana sovellusta. (GRASS development team, 2015) Laaja valikoima toimintoja ja mahdollisuus laajentaa sovellusta lisäosilla on tuonut GRASS GIS:n esimerkiksi arkeologian, merentutkimuksen, geofysiikan ja meteorologian sovellusalueiden käyttöön. (GRASS development team, 2013)

QGIS

QGIS:n (*Quantum GIS*) kehitys on saanut alkunsa tarpeesta paikkatietojärjestelmälle, jota on kenen tahansa helppo käyttää. Sitä kehitettiin alun perin Linux-alustalle, mutta nykyisin se on saatavilla sekä Windowsille, MacOSX:lle että Linuxille. QGIS sisältää vahvan GRASS GIS -integraation. Sovellukseen on nyttemmin kehitetty merkittävä määrä toiminnallisuuksia datan visualisoinnin lisäksi, ja QGIS:n laajentaminen lisäosien avulla on mahdollista. QGIS kuuluu OSGeo:n tukemiin projekteihin. (Steiniger & Hunter, 2013)

QGIS mahdollistaa useiden eri paikkatietoformaattien visualisoinnin. Se tukee sekä rasteri- että vektorikarttoja ja lukuisia eri tietokantoja. Karttojen tarkasteluun ja muokkaukseen on lukuisia mahdollisuuksia, esimerkiksi sijaintiin sidottujen kirjanmerkkien ja omien karttamerkintöjen tekeminen on helppoa. Geokoodaus ja GPS-paikannustietojen integrointi paikkatietoon on myös mahdollista QGIS:n avulla. Koska sovellus sisältää GRASS GIS -integraation, datan analysoiminen onnistuu myös kaikkien GRASS GIS:n moduulien avulla. QGIS voi toimia myös palvelimena karttojen jakamiseen useilla eri protokollilla. (QGIS development team, 2017)

gvSIG

Alun perin Espanjassa kehitetty gvSIG on saanut alkunsa tavoitteesta luoda avoin ja vapaa vaihtoehto kunnallisten organisaatioiden käyttämälle kaupalliselle ArcViewille. Projektilla on merkittävä taloudellinen tuki, ja nyttemmin sen käyttäjä- ja kehittäjäkunta on maailmanlaajuinen. Steinigerin ja Hunterin (2013) mukaan sen käyttöliittymä on yksinkertainen ja ohjelmisto on hyvin dokumentoitu.

Työpöytäkäyttöön tarkoitettu gvSIG Desktop mahdollistaa monien eri paikkatietoformaattien käsittelyn. Se tukee sekä vektori- ja rasterityyppisiä karttoja. So-

vellus tukee myös tulostettavien karttojen suunnittelua sekä tiedon muokkaamista ja analysointia yli 300 työkalun avulla. (gvSIG association, 2017) Laajennoksien kehittäminen on myös mahdollista, joskin Steiniger ja Hunter (2013) mainitsevat kehittäjille tarjotun dokumentaation olevan puutteellista. Lisäksi heidän mukaansa projektilla on mittava määrä riippuvuuksia eri C++- ja Java-kirjastoihin, joka osaltaan hankaloittaa sovelluksen laajentamista.

2.3.2. Suljettu lähdekoodi

Vaikka avoimen lähdekoodin sovelluksien määrä onkin kasvanut selkeästi viime vuosina, suljetuilla kaupallisilla paikkatietosovelluksilla on edelleen suuri käyttäjäkunta. Esimerkiksi ESRI:n (*Environmental Systems Research Institute*) markkinaosuus vuonna 2015 oli noin 43%. (Environmental Systems Research Institute, 2015) Vuonna 2016 ESRI:n tuotteita ja teknologiaa käytettiin noin 350 000 yrityksessä, valtiollisessa virastossa ja kansalaisjärjestössä. Yhteensä nämä tahot luovat toistasataa miljoonaa karttaa päivässä. (Helft, 2016)

ArcGIS

ESRI on markkinoita johtava paikkatietojärjestelmien toimittaja. Sen ArcGIS-tuoteperhe kattaa laajalti teollisuuden eri sovellusalueet, ja tuotteet pyrkivät vastaamaan erityisesti paikkatietojärjestelmien tehokäyttäjien tarpeisiin. (Longley ja muut, 2005, s. 165–167) Merkittävää on myös, että laajalti käytetty pääosin avoin shapefile-formaatti on alun perin ESRI:n kehittämä. Monet nykyiset avoimet ja suljetut paikkatietojärjestelmät tukevat tätä formaattia.

ESRI:n ArcGIS koostuu useista eri sovelluksista. Työpöytäkäyttöön tarkoitetun ArcGIS Desktopin lisäksi järjestelmään kuuluvat muun muassa yrityskäyttöön tarkoitetut palvelinmalliset taustapalvelut sekä pilvipalveluna toimiva ArcGIS Online. ArcGIS tukee lukuisia tiedostoformaatteja ja merkittävää määrää erilaisia työkaluja paikkatiedon käsittelyyn. Kuvien prosessointiin ArcGIS tarjoaa esimerkiksi koneoppimisen avulla kuvien ja alueiden luokittelun. Lisäksi esimerkiksi ortoilmakuvien vääristymien korjaaminen on mahdollista. Paikkatiedon käsittelyn automatisointi erilaisten skriptien avulla on myös mahdollista, ja ArcGIS:lle on saatavilla useita lisäosia. (Environmental Systems Research Institute, 2017)

MapInfo

MapInfo on Pitney Bowesin kehittämä paikkatietosovellus. Se sisältää kaikki paikkatietojärjestelmän perusominaisuudet: paikkatiedon visualisoinnin, analysoinnin ja muokkauksen. MapInfoa voidaan käyttää erilaisten karttojen luomiseen ja se tarjoaa edistyneen ja helppokäyttöisen käyttöliittymän. MapInfo tukee useita eri tietokantoja ja tiedostoformaatteja. (Pitney Bowes, 2017)

Myös kuvatiedostojen kuten taustakarttojen, ilmakuvien ja satelliittikuvien muokkaus ja analysointi onnistuvat MapInfolla. Vektori- ja rasterikarttoja voidaan käyttää yhdenaikaisesti lataamalla vektoridataa rasterikarttojen päälle. MapInfo on myöskin laajennettavissa MapBasicilla, joka on erityisesti MapInfoa varten kehitetty ohjelmointikieli. MapBasic mahdollistaa myös sovelluskehityksen yleisemmillä kielillä, kuten C ja C++. (Pitney Bowes, 2017)

GeoMedia

GeoMedia on Hexagon Geospatialin kehittämä paikkatietosovellus. Se tukee lukuisia eri paikkatietoformaatteja sekä tietokantoja, ja pyrkii tarjoamaan yhden näkymän monista eri tietolähteistä yhdisteltyyn dataan. GeoMedian analysointityökalut pyrkivät kattamaan suurienkin organisaatioiden tarpeet. Työpöytäsovelluksen lisäksi tuoteperheeseen kuuluu lisäksi palvelinohjelmistoa ja erilaisia mobiilisovelluksia. (Hexagon Geospatial, 2017)

Myös ESRI:n tietokantojen käyttäminen on mahdollista GeoMedian kautta. GeoMedian kohdeyleisöä ovat organisaatiot, joiden liiketoiminnan kannalta organisaation laajuinen paikkatietojärjestelmä on merkittävässä asemassa. Se ei kuitenkaan käytä omaa suljettua paikkatietoaineistoaan. GeoMedia toimii esimerkiksi OGC:n standardien mukaisten tietoaineistojen parissa. (Hexagon Geospatial, 2014)

3 Paikkatieto

Paikkatieto on keskeinen osa paikkatietojärjestelmiä ja -sovelluksia. Sillä pyritään kuvaamaan jonkin olion tai ominaisuuden sijainti maapallon pinnalla tai pinnan läheisyydessä. Sijainti voi olla absoluuttinen tai suhteellinen, ja sen esittämiseen käytetään useimmiten koordinaatteja. Äärellisten koordinaattijoukkojen avulla saadaan myös kuvattua kohteen muoto ja koko käyttäen geometrisiä primitiivejä. Paikkatieto on aina jollakin tapaa yksinkertaistettu esitys reaalimaailmasta, joten esimerkiksi kohteiden muotoa on tarpeen yleistää. Vaatimukset paikkatiedon tarkkuudelle ja laadulle riippuvat sovellusalueesta. (Heywood ja muut, 2006, s. 32–43)

Paikkatietokohteella tarkoitetaan abstraktia kuvausta jostakin reaalimaailman oliosta, joka on tietyssä maantieteellisessä paikassa. Jos tarkastellaan moottoripyöränavigaattoria paikkatietosovelluksena, sen kannalta oleellisia paikkatietokohteita ovat esimerkiksi tiet, risteykset ja nopeusrajoitukset. Purot, jyrkänteet ja suot eivät kiinnosta moottoripyöränavigaattorin käyttäjää, mutta maastossa liikkuvalle ne voivat olla hyvinkin oleellisia paikkatietokohteita.

3.1. Diskreetit oliot ja jatkuvat alueet

Maapallon maantiedon määrittelyyn paikkatiedon avulla on kaksi erilaista lähestymistapaa. Maapallon pinnan voidaan katsoa koostuvan pelkästä tyhjästä tilasta, jota diskreetit oliot (*discrete objects*) täyttävät. Näillä olioilla on tarkasti määritellyt rajat jotka määrittävät myös sijainnin. Diskreettien olioiden välillä on vain tyhjää tilaa, ja oliot ovat helposti laskettavissa. Diskreeteillä olioilla voidaan helposti kuvata niitä kohteita, joilla on selkeät rajat reaalimaailmassakin. Diskreeteistä olioista koostuva maailmankuva soveltuu kuitenkin huonosti monien luonnosta löytyvien kohteiden kuvaamiseen. Esimerkiksi joen kuvaaminen viivajoukolla jättää pois joen leveyden, mikä saattaa hyvinkin olla vesistönylitystä suunnittelevalle suunnistajalle oleellista tietoa. (Longley ja muut, 2005, s. 70–72)

Jos diskreeteillä olioilla pitäisi kuvata maastonmuotoja kuten mäkiä, laaksoja tai kuruja, olisi rajojen määrittely hankalampaa. Tällaiset maantieteelliset muodot eivät sovellu kuvattavaksi diskreeteillä olioilla. Niiden kuvaamiseen tarvitaan toista lähestymistapaa: jatkuvia alueita (*continuous fields*). Maailman voidaan tyhjän tilan sijaan katsoa koostuvan myös jatkuvasta alueesta, jonka sisällä rajallinen määrä muuttujia saa erilaisia arvoja sijainnista riippuen. (Longley ja muut, 2005, s. 72–74) Geoinformatiikan sanaston (2014) mukaan jatkuviin alueisiin voidaan viitata myös

nimellä paikkatietojatkumo.

Jatkuvalla alueella maantietoa kuvattaessa jokaiselle alueen pisteelle on siis määritelty yksi tai useampi muuttuja, jotka voivat olla esimerkiksi nominaalisia, ordinaalisia, intervalliasteikollisia tai suhdeasteikollisia. Muuttujan tyyppi riippuu mitattavasta suureesta, eikä jatkuvalla alueella ole mitään rajoitteita muuttujatyypin suhteen. (Longley ja muut, 2005, s. 72–74)

Jatkuvaa aluetta voitaisiin hyödyntää esimerkiksi säteilyn alueellisen voimakkuuden tallentamiseen ja esittämiseen. Tällöin jokaisella alueen pisteellä olisi säteilyn voimakkuutta kuvaavalle muuttujalle jokin arvo. Keskeistä on huomata, että vaikka paikkatiedon esitystapa olisi jatkuva alue, aineisto ei tietenkään voi sisältää tarkkaa säteilyn voimakkuutta jokaiselle alueen pisteelle. Tämä on seurausta alueen jatkuvuudesta; se sisältää äärettömän määrän pisteitä. Toisaalta säteilyn voimakkuutta mitattaessa ei voida myöskään mitenkään kattaa koko alueen kaikkia pisteitä. Tämä tarkoittaa sitä, että tallennetuista mittauspisteistä on kyettävä johtamaan jollakin tarkoitukseen sopivalla laskukaavalla säteilyn voimakkuutta kuvaava arvo mille tahansa jatkuvan alueen pisteelle.

Maastokartoissa keskeinen jatkuva alue on korkeustieto. Korkeusmallissa alueen jokaiselle pisteelle määritellään jokin korkeus, joka voi pisteiden välillä vaihdella sekä nopeasti että hitaasti. Tasaisella, jääkauden kuluttamalla alueella korkeusmuuttujan arvo vaihtelee hitaasti liikuttaessa pisteestä toiseen. Kallioisessa ja jyrkänteisessä maastossa jatkuvan alueen pisteiden arvot vaihtelevat nopeammin. (Longley ja muut, 2005, s.72–74) Vaikka korkeustieto on selkeä jatkuvan alueen sovelluskohde, esimerkiksi paperisilla maastokartoilla korkeus merkitään yleensä erityisillä korkeuskäyrillä. Korkeusmalleja esittelen tarkemmin kohdassa 3.5. Diskreeteillä olioilla ja jatkuvilla alueilla on vahva yhteys rasteri- ja vektorikarttoihin, joita esittelen tarkemmin kohdassa 3.4.

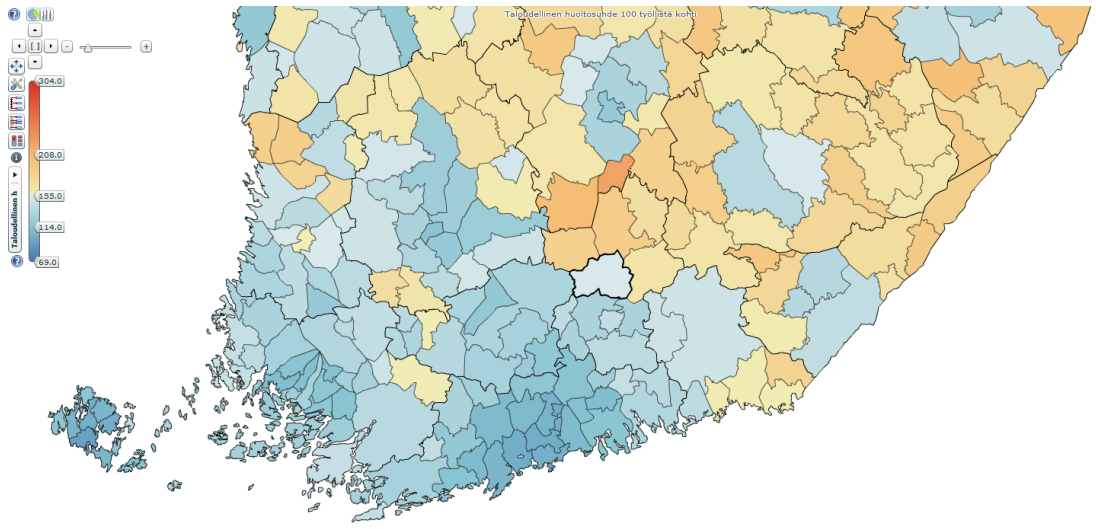
3.2. Sijainti ja geometria

Oliolla voidaan paikkatiedon yhteydessä tarkoittaa esimerkiksi linja-autoa, jokea tai peltoa. Kaikki edellä mainituista konkreettisista olioista sijaitsevat kolmiulotteisessa avaruudessa. Paikkatietoa varten niiden sijainti on usein tarpeen määritellä kaksiulotteisessa avaruudessa, kuten esimerkiksi kartalla. Jos joki halutaan kuvata kaksiulotteisessa avaruudessa, on ilmeistä, että sen sijainti ei ole mikään yksittäinen piste. Joki voi virrata useita satoja kilometrejä ennen päättymistään mereen. Joen kuvaamiseen on siis syytä käyttää esimerkiksi kokoelmaa pisteitä, joita yhdistää janat.

Toisaalta linja-auton sijainti voidaan hyvinkin esittää yksittäisenä pisteenä. Pellon kuvaamiseen kaksiulotteisella kartalla tarvitaan alue, eli monikulmio.

Edellä mainitut kolme paikkatiedon peruselementtiä *piste*, *viiva* ja *alue* ovat keskeisimmät tavat kohteiden geometrian esittämiseen. (Gatrell, 1991) Niitä kutsutaan myös sijaintiobjekteiksi (*spatial object*), sillä ne kuvaavat sijaintiin liittyviä ominaisuuksia. (Sanastokeskus TSK, 2014) Pisteen eli yksikäsitteisen paikan määrittelemiseen tarvitaan vain yhdet koordinaatit. Kaksiulotteisella kartalla nämä koordinaatit sisältävät kaksi arvoa: yksi kullekin kartan ulottuvuudelle aivan kuten matematiikassa käytetyissä karteesisissa koordinaatistoissakin. Viiva koostuu järjestetystä joukosta päätepisteitä, jotka yhdistetään janoilla. Alueen määrittelemiseen tarvitaan myöskin järjestetty joukko pisteitä. Alueen ero viivaan on se, että alueen määrittelevässä pistejoukossa myös joukon ensimmäinen ja viimeinen piste yhdistetään janalla. (Gatrell, 1991)

Linja-auto, joki ja pelto ovat konkreettisia olioita joiden sijainti voidaan havaita reaalimaailmassa. Paikkatietoa voidaan käyttää myös hyvinkin abstraktien käsitteiden yhteydessä. Erityisesti paikkatietoja hyödyntävien tieteenalojen tapauksessa paikkatietoon liitetyt kohteet ovat usein abstrakteja. (Longley ja muut, 2005, s. 40–44)



Kuva 3. Etelä-Suomen taloudellinen huoltosuhde 100 työllistä kohti vuonna 2010 visualisoituna paikkatiedon avulla. (Tilastokeskus, 2017a)

Kuvassa 3 on esitetty kuvakaappaus Tilastokeskuksen kartta-animaatiosovelluksesta. Sovellus mahdollistaa Tilastokeskuksen väestölaskenta-aineiston tarkastelun kartan avulla. Sovelluksen käyttämä aineisto on saatavissa myös PX-Web-

tietokannasta. (Tilastokeskus, 2017b) Tietokannassa muuttujiin liittyvä sijaintitieto on tallennettu käyttämällä kunnan nimeä tai kunnalle määriteltyä koodia. Tästä huomataan, kuinka paikkatietoon liittyvä sijainti voidaan ilmaista muutenkin kuin pelkästään koordinaattijärjestelmän määrittelemien koordinaattien avulla.

Sijainti voidaan ilmaista joko suorasti tai epäsuorasti. Suoralla sijainnilla tarkoitetaan koordinaatteja ja epäsuoralla sijainnilla taasen esimerkiksi osoitetta tai muuta vastaavaa yksikäsitteistä järjestelmää. Epäsuoran sijainnin yhteydessä voidaan puhua myös hilarakenteesta, jolla tarkoitetaan esimerkiksi aluejakoa, jossa jaon tuloksena syntyviin alueisiin voidaan viitata yksikäsitteisesti. Kuntajako on siis tällainen hilarakenne. (Sanastokeskus TSK, 2014)

Kuvassa 3 esitettyä huoltosuhteen visualisointia varten kartta-animaatiosovellus tarvitsee toki myös tarkat kuntarajat, eli koordinaattipisteiden avulla määritellyt alueet. Kuntarajoja ei kuitenkaan ole tarvetta tallentaa erikseen jokaisen väestönlaskentaan liittyvän muuttujan osalta. Epäsuoralla sijainnilla voidaan helposti viitata suoralla sijainnilla toisaalla määriteltyyn paikkaan ja muotoon.

Kuten Gatrell (1991) huomauttaa, pelkästään kaksiulotteinen sijainnin tai geometrian ilmaisu ei ole välttämättä riittävä. Esimerkiksi miehittämättömien, matalalla lentävien lennokkien navigointijärjestelmien tapauksessa keskeisin paikkatieto eli lennokin sijainti määritellään kolmiulotteisessa avaruudessa. Reittisuunnitelmaksi ei tällöin riitä pelkkä viiva kaksiulotteisessa tasossa. Olioilla on paikkatiedon suhteen kaksi keskenään erilaista ulottuvuutta: niiden topologian määräämä ulottuvuus sekä niitä ympäröivän avaruuden ulottuvuus. (Gatrell, 1991) Paikkatiedon yhteydessä topologiatiedolla tarkoitetaan paikkatietokohteiden välisiä sijaintisuhteita kuvaava tietoa. (Sanastokeskus TSK, 2014) Kuten matemaattisessa topologiassakin, topologiatiedossa olennaista on se, että nämä sijaintisuhteet säilyvät muuttumattomina jatkuvissa muunnoksissa.

3.3. Paikkatiedon tallentaminen

Paikkatiedon tehokas tallentaminen on ollut merkittävä haaste paikkatietojärjestelmien historiassa. Goodchildin (2009a) mukaan relaatiotietokantojen kehittyminen 1970-luvulla vei myös paikkatiedon tallentamista eteenpäin. Ennen relaatiotietokantoja paikkatiedon tallentaminen perustui räätälöityihin ja omaperäisiin ratkaisuihin, mikä tietenkin osaltaan vaikeutti paikkatietojärjestelmien laajamittaista kehitystä ja tiedonvaihtoa. Ensimmäiset relaatiotietokannat eivät kuitenkaan olleet täydellisiä ratkaisuja paikkatiedon tallentamiseen, joten suljetut, räätälöidyt ratkaisut säilyt-

tivät ainakin osin asemansa. Paikkatietoa tallennetaan usein myös karttojen muodossa. Karttoja käsittelen tarkemmin luvussa 4.

Aref ja Samet (1991) esittävät, kuinka paikkatiedon tallentaminen voidaan toteuttaa arkkitehtuurilla joka erottaa sijaintitiedon ja objektitiedon erillisiin tietovarastoihin. Sijaintitieto tallennetaan arkkitehtuurissa sellaisiin tietorakenteisiin, joihin voidaan kohdistaa tehokkaita sijaintitiedollisia algoritmeja ja erilaisia jatkuvia muunnoksia. Myöhempi oliotietokantaparadigman kehittyminen on tehnyt paikkatiedon tallentamisesta helpompaa ja joustavampaa, eivätkä tällaiset Arefin ja Sametin kuvailemat hybridiarkkitehtuurit ole enää välttämättömiä. (Goodchild, 2009a)

Paikkatiedon tallentaminen relaatiotietokantaan voidaan yksinkertaisimmillaan toteuttaa siten, että tietokannan tauluun lisätään tarvittavat sarakkeet paikkatietoa varten. Tämä voi tarkoittaa esimerkiksi kahden koordinaattisarakkeen lisäämistä pistesijainnin tapauksessa. Alueita tai viivoja tallennettaessa sarakkeita tarvittaisiin useampia vaihtelevissa määrin. Tämä tuo hyvin esille yhden paikkatiedon tallentamista koskevan tietokantojen suunnitteluhaasteen: paikkatietoon liittyvä sijainti on monimuotoista ja -ulotteista. Toinen keskeinen haaste paikkatiedon tallentamisessa on paikkatietoa tallentavaan tietokantaan kohdistuvien kyselyiden luonne. Paikkatietoa on usein tarpeen tarkastella ja analysoida sijaintitiedon implisiittisten ominaisuuksien pohjalta. Paikkatietokohteiden sijaintien keskinäiset suhteet ovat hyvä esimerkki tästä. (Samet, 1995)

Keskinäisten sijaintisuhteiden merkitys korostuu esimerkiksi tarkasteltaessa linja-autojen lippuvyöhykkeitä. Lippuvyöhykkeet voidaan helposti määritellä alueina. Ne ovat oleellisia silloin, kun linja-autojen reittejä ja pysäkkien välisiä hinnoitteluja määritellään. Vyöhykkeitä ei kuitenkaan ole syytä tallentaa paikkatietokantaan tie- ja pysäkkikohtaisesti, joten kunkin pysäkin ja tieosuuden vyöhyketieto on johdettava lippuvyöhykkeen aluetiedon, tien viivatiedon ja pysäkin pistetiedon pohjalta. Jos käsiteltävä alue on valtavan laaja, tällaisen tiedon johtaminen voi olla suunnattoman työlästä ja laskennallisesti vaativaa.

Samet (1995) mainitsee yhtenä esimerkkinä sijaintitietojen implisiittisistä suhteista teiden risteysalueet. Tietietokantaan on kohtuutonta tallentaa jokaisen tien osalta tieto kaikista risteävistä teistä, ja toisaalta risteävien teiden laskenta pelkästään kaikkien teiden viivatietojen pohjalta vaatii sunnattomasti aikaa ja laskentaresursseja. Tästä syystä paikkatiedon tallennukseen on ollut tarpeellista kehittää menetelmiä, jotka mahdollistavat tämänkaltaisiin kyselyihin vastaamisen nopeasti ja tehokkaasti.

Paikkatietoa varten on kehitetty lukuisia tietotyyppejä, jotka abstrahoivat se-

kä tietotyypin rakenteen että siihen kohdistuvat operaatiot. Tällaiset tietotyypit mahdollistavat paikkatietojen käytön osana relaatiotietokantojen attribuuttijoukkoa. Abstrakti tietotyyppi mahdollistaa esimerkiksi sijaintisuhteita käsittelevien kyselyiden kohdistamisen relaatiotietokantaan, jossa paikkatieto on vain yksi attribuutti muiden perustietotyyppien (kokonaisluku, liukuluku, totuusarvo ynnä muut) rinnalla. (Schneider, 2009)

Tärkeimpänä paikkatiedon tallennusmuotona voidaan Schneiderin (2009) mukaan pitää rakenteeseen perustuvia sijaintitietotyyppejä (*structure-based spatial data types*). Tällaiset rakenteeseen perustuvat tietotyypit tallentavat olion geometrian, eli sen rakenteellisen muodon ja ulottuvuuden. Nämä rakenteeseen perustuvat tietotyypit voidaan edelleen jakaa yksinkertaisiin (*simple*) ja monimutkaisiin (*complex*) tietotyyppeihin.

Aiemmin mainitut piste, viiva ja alue ovat sellaisenaan yksinkertaisia tietotyyppejä. Jos tarkastellaan esimerkiksi kahden sisäkkäisen alueen erotusta tai symmetristä erotusta, tuloksena syntyvä alue sisältää reiän. Tällaista alueiden erotusta ei voida esittää yksinkertaisen alueen avulla. Kuten Schneider (2009) mainitsee, yksinkertaiset rakenteeseen perustuvat tietotyypit eivät ole suljettuja geometristen operaatioiden suhteen. Yksinkertaisiin rakenteeseen perustuviin tietotyyppeihin kohdistetut operaatiot voivat siis tuottaa sijaintitiedon, joka on rakenteeltaan monimutkainen.

Reitinhaun näkökulmasta merkittävä rakenteeseen perustuva monimutkainen tietotyyppi on paikkaverkko (*spatial network*). Paikkaverkolla tarkoitetaan graafia eli verkkoa, jonka solmuille ja kaarille on määritelty geometriset sijainnit. (Schneider, 2009) Graafin matemaattinen määritelmä ei ota kantaa solmujen tai kaarien sijainteihin avaruudessa. Graafien ominaisuuksien analysointiin on kuitenkin kehitetty merkittävä määrä paikkatietosovellusten näkökulmasta mielenkiintoisia algoritmeja ja menetelmiä. Näistä tämän tutkielman kannalta oleellisena voidaan mainita reitinhaku, erityisesti painotetussa graafissa.

Paikkatiedon indeksointi on tärkeä keino paikkatiedon käytön tehostamiseen. Tavanomaisten tietokantojen indeksointitekniikoihin kohdistuu merkittävä määrä tutkimusta, mutta ne soveltuvat huonosti paikkatiedon indeksointiin. Paikkatiedon käsitellessä moniulotteisia avaruuksia siihen kohdistuvat kyselyt ja operaatiot vaativat indeksoinnilta lineaarisen järjestyksen toteutumisen. (Manolopoulos, Theodoridis, & Tsotras, 2009)

Paikkatiedon indeksointiin kehitetyt menetelmät indeksoivat joko moniulotteisia pisteitä tai moniulotteisia alueita. Moniulotteisuudella tarkoitetaan tässä tyypillisesti kaksi- tai kolmiulotteisuutta. Moniulotteisia pisteitä indeksoiviin menetelmiin

kuuluvat muun muassa Henrichin, Sixin ja Widmayerin (1989) LSD-puu sekä Nievergeltin, Hinterbergerin ja Sevcikin (1984) ruudukkotiedosto (*grid file*). Moniulotteisten alueiden indeksointiin soveltuvat esimerkiksi Guttmanin (1984) R-puu ja sen lukuisat variantit sekä Finkelin ja Bentleyn (1974) nelipuu (*quadtree*) variantteineen. (Manolopoulos ja muut, 2009)

Sekä pisteiden että alueiden indeksointiin kehitetyt tietorakenteet voidaan edelleen jakaa tilavetoisiksi (*space-driven*) ja datavetoisiksi (*data-driven*) rakenteiksi. Nievergeltin ja muiden (1984) ruudukkotiedosto on tilavetoinen tietorakenne, joka indeksoi moniulotteisia pisteitä. Se jakaa koko hakuavaruuden epäsäännölliseksi ruudukoksi ottaen huomioon datan jakauman. Kaikki tietorakenteeseen tallennettava data sijoittuu johonkin ruutuun. Ruuduista tallennetaan kaksisuuntaiset viitteet niihin sijoittuvaan dataan. Data itsessään sijoitetaan johonkin toissijaiseen säiliöön. Jos yksittäinen ruudukon osa tulee liian täyteen, hakuavaruutta jaetaan edelleen pienemmiksi osiksi.

R-puulla voidaan tallentaa sekä pisteitä että alueita. Se voidaan luokitella datavetoiseksi rakenteeksi. R-puu on tasapainotettu puu, jossa hakuavaruudessa toisiaan lähellä olevat elementit tallennetaan samaan alipuuhun. Toisiaan lähekkäin olevilla puuhun tallennetuille elementeille lasketaan niiden sijaintia rajaava suorakulmio, joka tallennetaan puussa seuraavaksi matalammalle tasolle. Edelleen puussa matalammalle tallennetaan suorakulmio, joka sulkee sisäänsä pienempiä elementtejä sisältävät suorakulmiot. Puurakenteelle tyypillisesti syvemmälle edettäessä hakuavaruus supistuu, kunnes haluttu data löytyy. Tällaisesta rakenteesta on helppo hakea esimerkiksi jostakin pisteestä tietyllä etäisyydellä sijaitsevat kohteet. (Guttman, 1984)

3.4. Rasterit ja vektorit

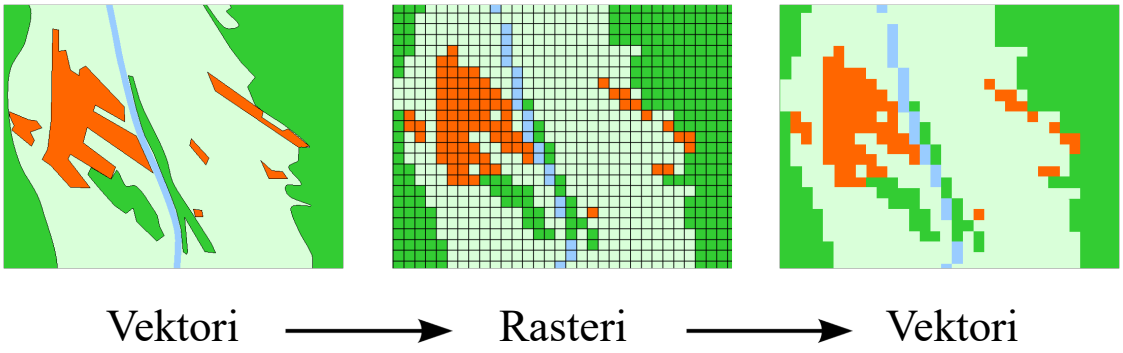
Kohdassa 3.1 käsittelemilläni diskreeteillä olioilla ja jatkuvilla alueilla on läheinen yhteys paikkatiedon tallennusmenetelmiin. Paikkatietoa voidaan tallentaa esimerkiksi karttoja varten joko rasterimuotoisena tai vektorimuotoisena. Rasterilla tarkoitetaan ruudukoitua aluetta, jonka yksittäiselle ruudulle on määritelty jotakin ominaisuutta kuvaavan suureen arvo. (Longley ja muut, 2005, s. 74–76) Yksittäisen ruudun sisällä arvo ei siis vaihtele rasterissa, mutta toisaalta vierekkäisten ruutujen arvot voivat vaihdella. Esimerkiksi valokuvat tallennetaan tyypillisesti rasterina: kuva on jaoteltu pikseleihin, ja yksittäisellä pikselillä on tietty väriarvo.

On selvää, että rasterimuodossa paikkatiedon tarkkuus kärsii helposti. Rasteri

ei sisällä tietoa vaihtelusta yksittäisen ruudun sisällä, joten ruutujen koolla on huomattava vaikutus lopullisen esityksen tarkkuuteen. Rasteria muodostettaessa täytyy määritellä, kuinka ruudulle lasketaan sitä kuvaava arvo. Jos useampi paikkatietokohde osuu yksittäisen ruudun alueelle, voidaan näistä kohteista valita esimerkiksi se, jonka pinta-ala ruudun sisällä on suurempi. Toisaalta voidaan valita vaikkapa ruudun keskipisteessä sijaitseva paikkatietokohde rasterin arvoksi. (Longley ja muut, 2005, s. 74–76)

Rasteri soveltuu erityisesti tilanteisiin joissa paikkatietokohteiden rajat ja yksittäiset pisteet eivät ole hyvin määriteltyjä. Vektorimuotoinen data määrittelee sen sijaan rajat ja pisteet tarkasti, joten sillä on selkeä yhteys diskreetteihin olioihin. Vektoriesityksessä tallennetaan kohteita kuvaavat geometriset muodot siten, että niiden sijainnit on määritelty jossakin koordinaatistossa. Tämä mahdollistaa kohteiden äärimmäisen tarkan asettelun, eikä rasteriesityksessä tapahtuvaa yksittäisen ruudun sisäistä häviötä tapahdu. (Galati, 2006, s. 29–35)

Galati (2006, s. 31) mainitsee rasterin etuna sen yksinkertaisuuden. Rasteri on rakenteeltaan kuten taulukko ja siihen voidaan usein soveltaa kuvanprosessoinnista tuttuja algoritmeja. Rasterin käsittely on ohjelmistokehityksen näkökulmasta yksinkertaisempaa ja sen vaatima laskentateho on usein pienempi kuin vektorimuotoisen esityksen. Toisaalta jos lähdeaineisto on vektorimuotoista, sen rasterointi voi olla laskennallisesti raskasta.



Kuva 4. Muunnos vektorista rasteriksi ja takaisin vektoriksi. (Wikimedia Commons, 2007)

Vektorien ja rasterien välillä voidaan suorittaa muunnoksia, joskin tämä aiheuttaa väistämättä epätarkkuuksia lopputuloksessa. Kuvassa 4 on esitetty, miten vektorit ja rasterit eroavat toisistaan. Tässä rasteriin on valittu suuri ruudukkokoko, jotta eroavaisuudet korostuvat. Vasemmalla kuvattu vektorimuotoinen kartta voisi esittää

vaikkapa jotakin aluetta, jonka läpi virtaa joki. Oranssi alue on peltoa, ja vihreistä alueista tummempi kuvaa metsää vaaleamman alueen kuvatessa avoimempaa maastoa. Rasterimuotoon muunnettaessa huomataan erityisesti, että alueen läpi virtaava joki ei ole enää yhtenäinen. Ruudukon yksittäisten ruutujen leveys ylittää joen leveyden, joten tietyssä pisteessä joen päällä sijaitsevat rasteriesityksen ruudut saavat arvokseen avoimen maaston. Tällöin esimerkiksi rasteriesitykseen kohdistettu reitinhaku voisi löytää kuivalla maalla kulkevan reitin joen yli, vaikka todellisuudessa sellaista ei olekaan.

Kuvan 4 vektorikartan muunnos rasteriksi havainnollistaa hyvin muunnoksen mukanaan tuomia ongelmia. Jos karttaan sovellettava reitinhakualgoritmi sallii diagonaalisen etenemisen ruudukossa, joen ylityspisteitä löytyy entistä enemmän. Koska joki ei seuraa millään tapaa ruudukon akseleita, se ei säily yhtenäisenä muunnettaessa rasterimuotoon. Muunnettaessa kartta edelleen rasterista vektoriksi huomataan, että syntyvä vektoriesitys poikkeaa merkittävästi alkuperäisestä vektorikartasta. Kartta on ikään kuin pikselöitynyt. Tämä johtuu siitä, että muunnos rasterista vektoriksi on tehty yksinkertaisesti muuttamalla ruudukon ruutujen reunaviivat suoraan monikulmioiden reunaviivoiksi. Tässä voitaisiin kuitenkin esimerkiksi interpoloida reunaviivat siten, että ne eivät seuraa täsmällisesti ruudukon rajoja. Joka tapauksessa rasterista ei voida mitenkään palauttaa alkuperäistä vektoriesitystä täsmällisesti.

Useat paikkatietojärjestelmissä käytetyt algoritmit soveltuvat sekä vektoreille että rastereille. Toisaalta osa esimerkiksi vektoriesityksille soveltuvista algoritmeista ei ole sovellettavissa rastereihin ilman muunnosta vektoriksi. Kuten yleensäkin, valinta rasteri- ja vektoriesitysten välillä riippuu siitä, miten paikkatietoa aiotaan käyttää.

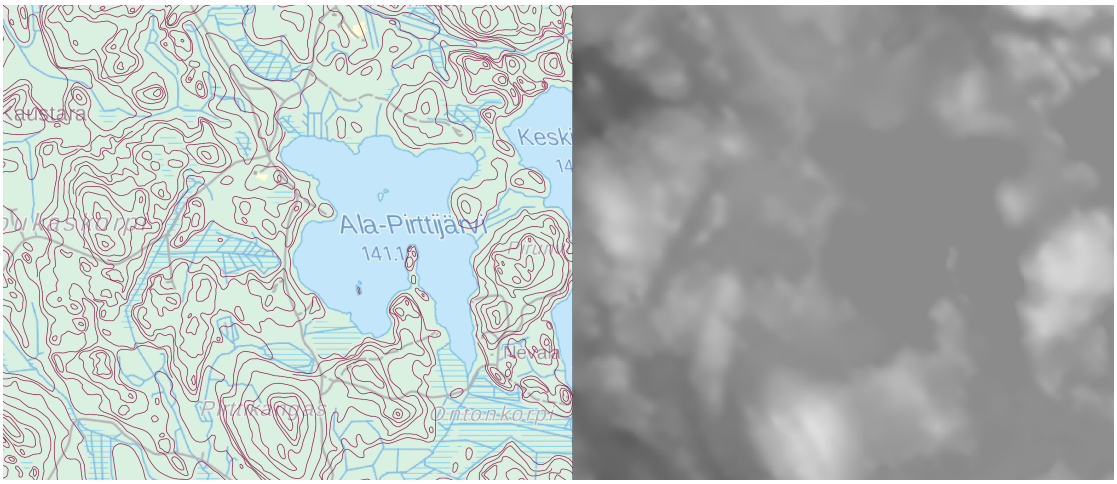
3.5. Korkeusmallit

Maastokartoilla reitinhakua toteutettaessa eräs merkittävä paikkatiedon ilmentymä on korkeusmalli. Korkeusmalli kuvaa maaston korkeutta jostakin kiinnitetystä tasosta, esimerkiksi meren pinnasta. Maastokartoissa korkeus kuvataan tyypillisesti korkeuskäyrillä. Korkeuskäyrä on paikkatietokohteena suljettu viiva. Tähän viivaan liittyy tieto siitä, mitä korkeutta viiva kuvaa. Ideaalitilanteessa tämän viivan jokaisessa pisteessä korkeus on sama, mutta käytännössä korkeuskäyriä joudutaan usein yleistämään. (Van Kreveld, Nievergelt, Roos, & Widmayer, 1997, s. 37–39)

Korkeusmalleja varten tietoa voidaan kerätä erilaisilla kaukokartoituslaitteilla. Mikäli tarkoituksena on mitata maaston korkeutta, tarvitsee kaukokartoituslaitteen

kyetä erottelemaan jollakin tapaa esimerkiksi puuston latvat ja rakennusten katot maan pinnasta. Maaston korkeutta voidaan toki arvioida myös maastosta käsin erilaisin mittauslaittein. Esimerkiksi GPS-paikantimilla saadaan mitattua korkeus merenpintaan tai muuhun mielivaltaiseen tasoon nähden. Nykyisin käytetään erityisesti LiDAR-teknologiaa (*Light Detection And Ranging*) tarkkojen korkeusmallien luomiseen. LiDAR:n avulla korkeusmalleja voidaan luoda jopa kahden metrin resoluutiolla suuriltakin alueilta. (Wilson, 2012)

Wilsonin (2012) mukaan yleisin korkeusmallin tallennusmuoto on neliöruudukoitu korkeusmalli. Se vastaa rasteria, jossa yksittäisellä ruudulla on jokin tietty korkeusarvo. Vastaavasti voitaisiin todeta, että edellä mainitut korkeuskäyrät ovat korkeusmallin vektoriesitys. Kuten kohdassa 3.4 mainitsin, vektorit soveltuvat huonosti tilanteisiin, joissa alueiden väliset rajat eivät ole selkeitä. Näissä tilanteissa rasteriesityksellä voidaan muodostaa tarkempi malli alueista.



Kuva 5. Korkeuskäyrät ja ruudukoitu korkeusmalli samasta alueesta. (Maanmittauslaitos, 2017a)

Kuvassa 5 on esitetty erään alueen korkeusmalli viiden metrin korkeuskäyrien ja ruudukoidun korkeusmallin avulla. Molemmat kartat ovat peräisin Maanmittauslaitoksen avoimien aineistojen tiedostopalvelusta. Vasemmalla esitetyn taustakartan päälle on piirretty korkeuskäyrät tumman ruskealla värillä. Korkeuskäyrät ovat osa Maanmittauslaitoksen maastotietokantaa. Jokaiseen korkeuskäyrään liittyy tieto kyseisen käyrän merkitsemästä korkeudesta, ja lähtökohtaisesti sisäkkäisten korkeuskäyrien välinen korkeusero on viisi metriä. Yksittäisten käyrien korkeuksia ei ole tähän kuvaan merkitty tilan säästämiseksi. Korkeuskäyriä tarkastelemalla saa hyvän yleiskäsityksen maaston muodoista, mutta kuten kuvastakin käy ilmi, korkeuskäyrät

ovat merkittävä yleistys maaston todellisista muodoista.

Kuvassa 5 oikealla esitetty ruudukoitu korkeusmalli on ruudukkokooltaan 10 m × 10 m. Mallin korkeustiedon tarkkuus on 1,4 metriä. Mittakaavasta johtuen tämän rasteriesityksen ruudut eivät erotu ihmissilmälle kuten kuvan 4 tapauksessa. Korkeus on visualisoitu esittämällä rasteri mustavalkoisena kuvana, jossa korkeusarvot on muunnettu pikseliarvovälille $[90, 200]$. Korkeuden kasvaessa kuvan pikselit muuttuvat siis vaaleimmiksi. Vaikka muunnos sopivalle pikseliarvovälille lisää kontrastia kuvaan, näyttää se silti suhteellisen tasaiselta ihmissilmälle. Koneellisesti luettuna korkeusmallin arvojen vaihtelu on kuitenkin selkeästi helpommin havaittavissa.

Ruudukoidulla korkeusmallilla voidaan myös interpoloida mille tahansa pisteelle korkeus laskemalla esimerkiksi etäisyyksien mukaan painotettu keskiarvo pistettä ympäröivien ruutujen korkeusarvoista. Interpolointi on usein tarpeen myös korkeusmallia muodostettaessa, sillä mittauspisteet eivät välttämättä noudata ruudukointia. Mittaustiheyttä voidaan muokata sen mukaan, kuinka vaihtelevaa maastonkorkeus on. Tällöin laajoille tasaisille alueille voidaan interpoloida korkeusarvoja. Korkeuskäyristä yksittäisen pisteen korkeuden määrittäminen vaatii myös merkittävästi interpolointia, sillä korkeus on määritelty alueittain tietyllä intervallilla. (Van Kreveld ja muut, 1997, s. 46–53) Kuvan 5 tapauksessa tämä intervalli on viisi metriä.

Esimerkiksi Chaplot ja muut (2006) ovat tutkineet eri interpolaatiotekniikoita ja niiden vaikutusta luotavan korkeusmallin laatuun. Keskeistä heidän havainnoissaan on, että käytetyn interpolaatiotekniikan merkitys vähenee mittauspisteiden tiheyden kasvaessa. Nykyiset kaukokartoituslaitteet vähentävät interpolaation tarvetta, joten nykyiset korkeusmallit ovat laadullisesti hyviä.

4 Kartat

Kohdassa 3.3 käsittelin paikkatiedon tallentamista yleisesti. Kuten todettua, paikkatieto voi liittyä mihin tahansa maapallon pinnalla tai sen läheisyydessä sijaitsevaan kohteeseen. Paikkatiedosta ei kuitenkaan kaikissa tilanteissa ole mielekästä muodostaa karttoja. Kartat ovat silti keskeisin paikkatiedon sovelluskohde ja paperikartalla on pitkät perinteet paikkatiedon tallennus- ja visualisointimuotona. Vaikka digitaaliset kartat ovat saavuttaneet valtavan suosion ja käyttäjäkunnan, analogisilla paperikartoilla on silti paikkansa. Esimerkiksi valtamerinavigoinnissa käytetään usein paperikarttoja digitaalisten rinnalla siltä varalta, että digitaaliset järjestelmät lakkaavat toimimasta.

Tässä luvussa käsittelen karttoja ja maapallon maantieteen karttamuotoisen esityksen haasteita. Reitinhakualgoritmeja karttoihin sovellettaessa tulee huomioida useita lopputuloksen tarkkuuteen vaikuttavia seikkoja ja karttojen ominaisuuksien tunteminen on oleellinen osa reitinhakualgoritmien valintaa ja tulosten arviointia. Esimerkiksi reitin tarkan pituuden laskeminen edellyttää valmiiksi muodostetun kartan tapauksessa kartan projektion ja koordinaattijärjestelmän tuntemista. Tyypillisesti paikkatietoaineistot sisältävät tiedon käytetystä projektiosta ja koordinaattijärjestelmästä. Paikkatietojärjestelmät käyttävät tätä tietoa edelleen esimerkiksi sovellettaessa paikkatietokohteisiin erilaisia geometrisia operaatioita. (Heywood ja muut, 2006, s. 44–46)

4.1. Karttaprojektiot

Karttaprojektio kuvaa, kuinka pisteet kolmiulotteisessa avaruudessa projisoidaan kaksiulotteiselle tasolle. Maapallo ei ole nimestään huolimatta muodoltaan pallo, vaan muistuttaa pikemminkin sferoidia eli litistynyttä palloa. Maapallon ympärysmitta on suurempi päiväntasaajalta mitattuna kuin pystysuunnassa napojen kautta mitattuna. Maapallon muotoa mallinnetaan matemaattisesti esimerkiksi erilaisilla pyörähdysellipsoideilla. Tämä muoto estää maapallon pinnan kuvaamisen tarkasti kaksiulotteisella tasolla. Kaarevaa viivaa ei voida kuvata yksiulotteisella tasolla ilman vaihtelevaa mittakaavavirhettä. Vastaavasti pyöreää pintaa ei voida kuvata kaksiulotteisella tasolla ilman vääristymiä mittakaavassa. Lisäksi maapallo on kooltaan niin suuri, että käytännön sovelluksiin projektiota tarvitaan myös skaalaamiseen eli mittakaavan muuttamiseen. Karttaprojektioissa keskeinen haaste on niiden aiheuttamat vääristymät. Projektioita on kehitetty useita ja kaikilla on eri käyttötarkoi-

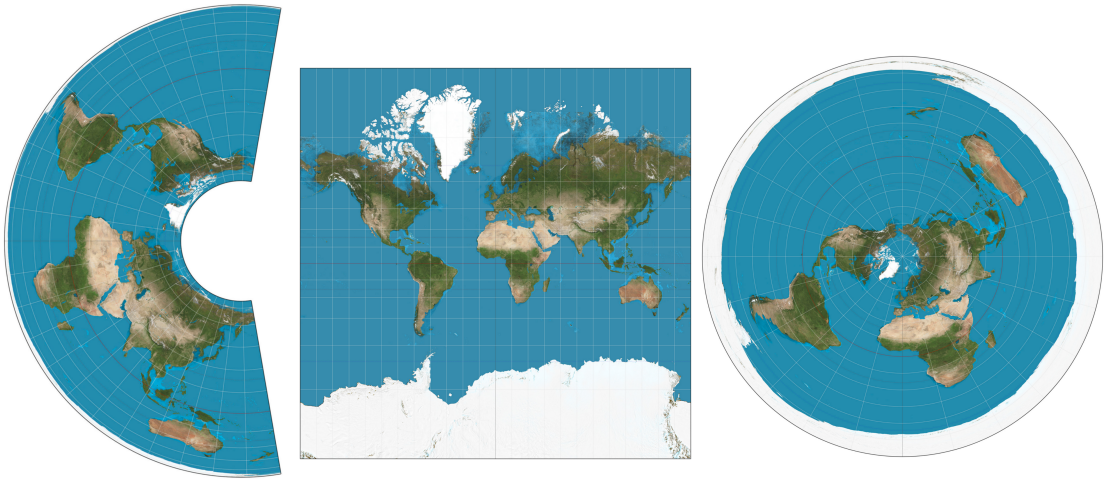
tuksensa. Soveltuvan projektion valinta on tärkeää, jotta paikkatietoa analysoidessa saavutetaan mielekkäitä tuloksia. (Panigrahi, 2014, s. 61–62)

Matemaattisesti karttaprojektion yleinen määritelmä on yksinkertainen. Se on mikä tahansa funktio f, joka kuvaa maapallon pallokoordinaatiston pisteen (ϕ, λ) kaksiulotteisen tason pisteeksi (x, y), eli $(x, y) = f(\phi, \lambda)$. Tässä ϕ on leveysaste ja λ pituusaste pallokoordinaatistossa. Vastaavasti käänteisprojektio f^{-1} kuvaa kaksiulotteisen tason pisteen pallokoordinaatistossa: $(\phi, \lambda) = f^{-1}(x, y)$. (Panigrahi, 2014, s. 62) Projektiota muodostettaessa tarvitaan myös maapallon datumi (*datum*), joka on koordinaatiston origon, mittakaavan ja orientaation määrittelevä parametrijoukko. (Sanastokeskus TSK, 2014) Datumi siis määrittelee, miten projektiopinta tulee kiinnittää projisoituun kappaleeseen, tässä tapauksessa maapalloon.

Vaikka tämä yleinen määritelmä on yksinkertainen, funktio f on joidenkin projektioiden osalta hyvinkin monimutkainen. Esimerkiksi Bugayevskiy ja Snyder (1995) esittelevät kirjassaan erilaisten projektioiden ja käänteisprojektioiden laskentakaavoja. Paikkatietojärjestelmät osaavat tyypillisesti käsitellä useita eri projektioita ja käänteisprojektioita. Projektioiden manuaalinen laskenta ei siis yleensä ole tarpeen, mutta kuten mainittua, niiden ominaisuuksien tunteminen on oleellista.

Soveltuvan projektion valintaa voidaan helpottaa luokittelemalla projektiot niiden tiettyjen ominaisuuksien mukaan. Vaikka matemaattinen malli mahdollistaa teoriassa rajattoman määrän erilaisia projektioita, niillä on kuitenkin yhteisiä kartografisia, geometrisia ja fyysisiä ominaisuuksia. Kartalla mitattavissa olevia perussuureita on Panigrahin (2014, s. 74–76) mukaan kolme: pinta-ala, muoto ja etäisyys. Karttaprojektiot aiheuttavat väistämättä vääristymiä kahteen näistä perussuureista. Toisaalta esimerkiksi Galati (2006, s. 121) erittelee nämä projektion vääristämät perussuureet vielä tarkemmin: pinta-ala, kulma, muoto, etäisyys ja suunta. Perussuureiden jaottelusta riippumatta voidaan kuitenkin todeta, että jos projektio kuvaa yhden perussuureen tarkasti tasokoordinaatistossa, muut perussuureet vääristyvät. Panigrahin (2014, s. 74–76) jaottelua noudattaen projektiolla voidaan siis esimerkiksi kuvata maapallon kohteiden pinta-alat tarkasti muotojen ja etäisyyksien vääristyessä.

Kuvassa 6 on esitetty kolme erilaista projektiota maapallon pinnasta. Kuvan vasemmassa reunassa esitetty Albersin projektio on oikeapintainen projektio (*equal area projection* tai *homolographic projection*). Oikeapintaisessa projektiossa eri alueiden pinta-alat esiintyvät oikeassa suhteessa. Projektion mittakaava kuitenkin vaihtelee eri suunnissa ja alueiden muodot vääristyvät merkittävästi. Kuten kuvasta 6 on nähtävissä, esimerkiksi Australian muoto vääristyy huomattavasti. Albersin projek-

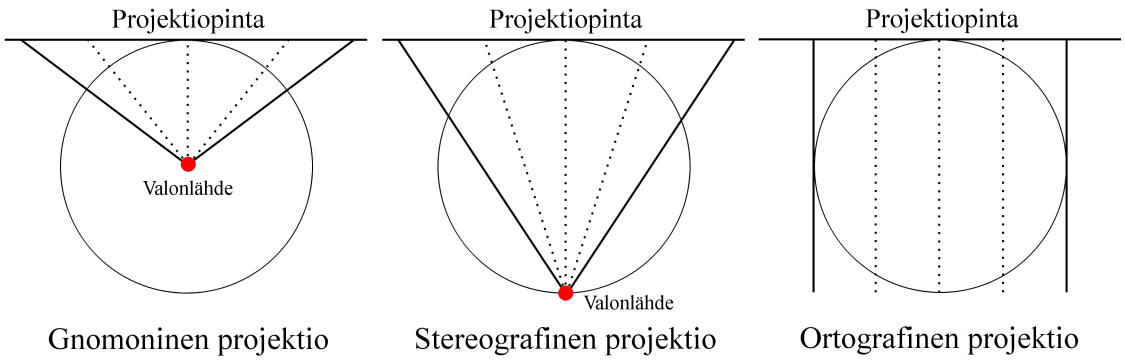


Kuva 6. Esimerkit projektioista jotka säilyttävät (vasemmalta oikealle) pinta-alan, muodon ja etäisyyden. (Wikimedia Commons, 2009)

tio hyödyntää kahta standardileveyspiiriä (*standard parallel*). Näiden standardileveyspiirien välille jäävällä osuudella muotojen vääristyminen on vähäistä. (Panigrahi, 2014, s. 75–82) Kuvassa 6 esitetyn Albersin projektion standardileveyspiirit ovat 20° ja 50° pohjoista leveyttä.

Kuvan 6 keskimmäinen kartta on muodostettu Mercatorin projektiolla. Mercatorin projektio on oikeakulmainen projektio (*orthomorphic projection* tai *conformal projection*), eli se kuvaa ilmansuuntien väliset kulmat täsmällisesti. Tämän ominaisuuden johdosta sen siis voidaan katsoa säilyttävän kohteiden muodon. Englanninkielisen nimensä mukaisesti se on rinnastettavissa matemaattiseen konformikuvaukseen. Tällaisessa kuvauksessa mittakaava vaihtelee kartan alueella, mutta se on mielivaltaisesti valitusta pisteestä jokaisen suuntaan sama. Tällaiset oikeakulmaiset projektiot soveltuvat hyvin käytettäväksi esimerkiksi merenkulussa. (Panigrahi, 2014, s. 75–82) Kuten kuvasta 6 nähdään, Mercatorin projektiossa napoja lähestyttäessä alueiden pinta-alat kasvavat huomattavasti. Jos verrataan kuvassa esitettyä Albersin projektiota ja Mercatorin projektiota, esimerkiksi Grönlanti on Mercatorin projektiossa huomattavasti suurempi Eurooppaan verrattaessa.

Kuvassa 6 oikeanpuolimmainen kartta hyödyntää oikeakeskipituista tasoprojektiota (*azimuthal equidistant projection*). Projektion keskipisteestä mitattuna etäisyydet pysyvät oikeassa suhteessa muiden pisteiden todellisiin etäisyyksiin. Kuvan 6 oikeakeskipituisessa tasoprojektiossa on yksi keskipiste, pohjoisnapa. Galatin (2006, s. 128) mukaan näitä keskipisteitä voi olla joissakin tapauksissa myös kaksi.



Kuva 7. Valonlähteen sijainti eri projektiossa. (Panigrahi, 2014, s. 77 mukaillen)

Projektioita voidaan luokitella myös sen perusteella mistä pisteestä datumiin nähden projisointi on tehty. Tyypillisesti tätä pistettä havainnollistetaan valonlähteen avulla. Jos maapallon pinta olisi läpikuultava, maapallon sisälle asetettu valonlähde muodostaisi pinnan läpi kulkiessaan kuvan maapallon tangentille asetetulle tasolle. (Panigrahi, 2014, s. 76–77) Kuvassa 7 on esitetty, miten punaisella ympyrällä merkityn valonlähteen sijainti vaikuttaa siihen, miten maapallon pinnalla sijaitsevat kohteet piirtyvät maapallon tangentille asetetulle tasolle eli projektiopinnalle.

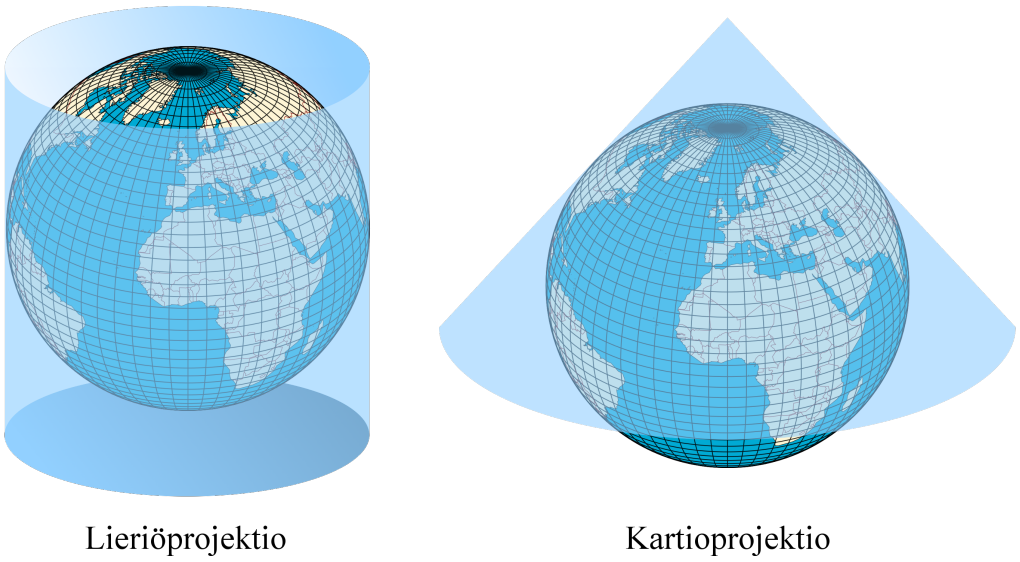
Kuvan 7 vasemmassa reunassa on esitetty gnomoninen projektio. Tässä projektiossa valonlähde on asetettu maapallon keskipisteeseen. Tällöin kahden maapallon pisteen välinen lyhin reitti kuvautuu projektiopinnalle suorana viivana. Kuten kuvastakin on arvioitavissa, projektion vääristymä on vähäisintä projektiopinnan ja maapallon pinnan leikkauspisteessä. (Panigrahi, 2014, s. 77)

Kuvan 7 keskimmäinen projektio on esimerkki stereografisesta projektiosta. Stereografisessa projektiossa valonlähde on sijoitettu pallon pinnalle, esimerkiksi jommallekummalle navalle. Valonlähteen sijaintipistettä lukuun ottamatta kaikki pallon pinnan kohteet kuvautuvat projektiopinnalle. Tällainen kuvaus on konforminen ja se soveltuu hyvin esimerkiksi maapallon trooppisten alueiden karttoihin. (Panigrahi, 2014, s. 77)

Kuvassa 7 oikeanpuolimmainen projektio eroaa kahdesta muusta kuvan projektiosta siten, että siinä valonlähde ei ole näkyvissä. Tällaisessa ortografisessa projektiossa valonlähteen ajatellaan olevan äärettömyydessä. Sitä kutsutaankin myös syvän avaruuden (*deep space*) projektioksi. Valonsäteet tulevat projektiopinnalle sen normaalien suuntaisesti. Tämä projektio soveltuu erityisesti satelliittikuvien projisoimiseen. (Panigrahi, 2014, s. 77)

Projektioita voidaan edelleen luokitella projektiopinnan muodon mukaan. Pro-

jektiopinnan ei välttämättä tarvitse olla yksinkertainen taso, kuten esimerkiksi kuvassa 7 on esitetty. Projektiopintana voidaan käyttää myös sylinteriä tai kartiota. Erilaisilla projektiopinnoilla on ominaisuuksia, jotka vaikuttavat vääristymien muodostumiseen. Projektiopinnalle muodostuu maapallon leveys- ja korkeuspiireistä hiusristi (*graticule*) jonka rakenne vaihtelee projektiopinnasta riippuen. (Panigrahi, 2014, s. 78–79)



Kuva 8. Lieriö- ja kartioprojektiot. (Wikimedia Commons, 2013; Galati, 2006, s. 123 mukaillen)

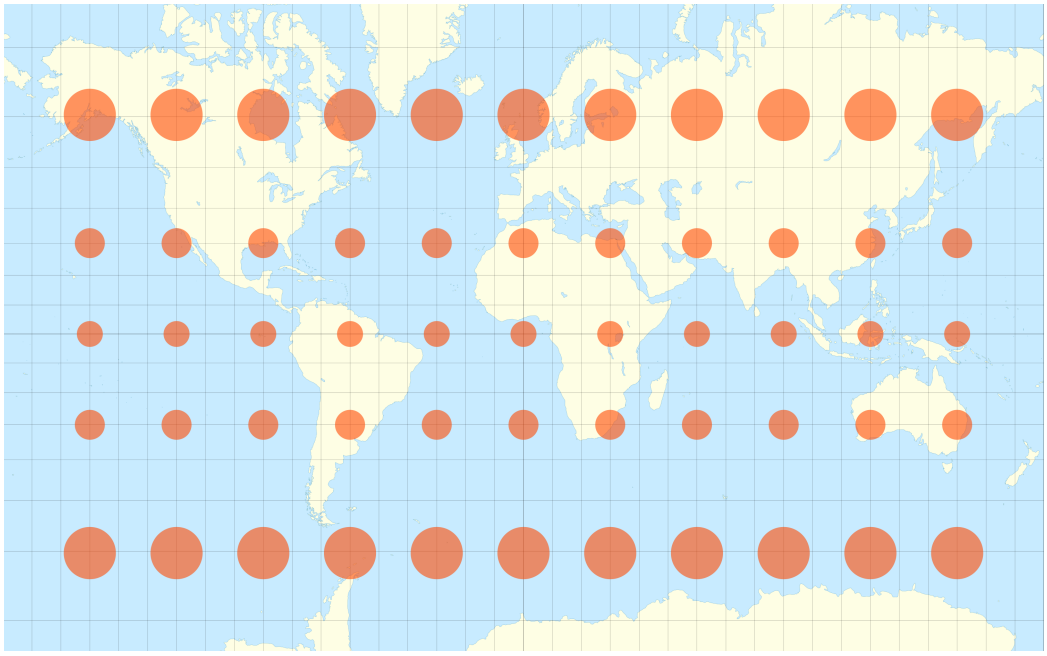
Kuvassa 8 on esitetty kaksi vaihtoehtoista projektiopintaa. Vasemman puoleisessa lieriöprojektiossa maapallon pinnalle piirretyt leveys- ja pituuspiirit kuvautuvat projektiopinnalle suorina viivoina. Leveys- ja pituuspiirit leikkaavat toisensa siis suorissa kulmissa, mutta projektion mittakaava muuttuu huomattavasti lähestyttäessä napoja. Kuvassa esitetyn kaltaisella lieriön sijoittelulla päiväntasaaja kuvautuu oikeassa mittakaavassa lieriön pinnalle. (Galati, 2006, s. 122–124)

Kuvan 8 oikean puolimmaisessa projektiossa projektiopinta on kartio. Tällaisessa projektiossa korkeuspiirit kuvautuvat suoriksi viivoiksi siten, että viivojen välinen kulma on sama kaikkialla. Leveyspiirit taasen kuvautuvat samankeskisiksi pyöreiksi kaariksi. Projektion keskeinen ominaisuus on se, että minkä tahansa kahden korkeuspiirin välinen kulma on projektiossa pienempi kuin niiden todellinen kulma maapallon leveyspiireillä mitattuna. (Galati, 2006, s. 122–124)

On tärkeää huomata, että projektiopinta voitaisiin kuvassa 8 asetella maapallon ympärille mihin tahansa asentoon. Lieriötä voitaisiin esimerkiksi kääntää siten, että

se leikkaisi maapallon millä tahansa korkeuspiirillä. Tällöin myös maapallon pohjois- ja etelänavat sijaitsisivat projektiopinnan leikkauksella. Tämä johtaa siihen, että valittu korkeuspiiri kuvautuu oikeassa mittakaavassa lieriön pinnalle. Toisaalta projektiopinnan kokoa voitaisiin myös pienentää siten, että esimerkiksi kuvassa 8 esitetty lieriö leikkaisi maapallon pinnan kahdella eri leveyspiirillä päiväntasaajan sijaan.

Panigrahi (2014, s. 79–80) mainitsee kolme eri luokittelua projektiopinnan asettelulle: ekvatoriaalinen (*equatorial* tai *normal*), kalteva (*oblique*) tai transversaalinen (*transverse* tai *polar*). Kuvassa 8 esitetty lieriöprojektio on ekvatoriaalinen, sillä projektiopinta on kohtisuorassa päiväntasaajan muodostamaan tasoon nähden. Transversaalisessa tapauksessa etelä- ja pohjoisnavat sijaitsevat projektiopinnan leikkauksella ja projektiopinta on yhdensuuntainen päiväntasaajan muodostaman tason kanssa.



Kuva 9. Tissot'n indikaattorit Mercatorin projektiossa. (Wikimedia Commons, 2008)

Projektioiden vääristymien havainnollistamiseen käytetään tyypillisesti Tissot'n indikaattoreita. Nicolas Auguste Tissot kehitti kyseiset indikaattorit vuonna 1881 karttojen projektiovääristymien analysointia varten. Tissot'n indikaattorit ovat projisoitavan pallon tai ellipsoidin pinnalle piirrettäviä ympyröitä, jotka projektion seurauksena muuttavat muotoaan. Mikäli ellipsoidin pinnalle piirretty ympyrä kuvautuu konformistisesti, se säilyttää muotonsa projektiossa. Vastaavasti ympyrän muodon muuttuessa projektiossa ellipsiksi kyseisellä alueella tapahtuu vääristymä, jonka voimakkuutta ympyrän vääristyminen kuvaa. Lisäksi indikaattoreihin kuuluu ruu-

dukko, jonka viivojen leikkauspisteisiin ympyrät kiinnitetään. (Galati, 2006, s. 133–135)

Kuvassa 9 näkyy, kuinka Tissot'n indikaattorit käyttäytyvät Mercatorin projektiossa. Ympyröiden muoto säilyy, mutta niiden koko vaihtelee. Tästä on hyvin nähtävissä Mercatorin projektion yksi keskeinen ominaisuus: päiväntasaajalla sijaitsevat kohteet kuvautuvat huomattavasti pienemmiksi kuin lähellä napoja sijaitsevat kohteet. Lisäksi Tissot'n indikaattoreista voidaan arvioida Mercatorin olevan todennäköisesti konformikuvaus, sillä ympyrät säilyttävät muotonsa. Pelkkä indikaattorien visuaalinen tarkastelu ei tietenkään anna täyttä varmuutta projektion ominaisuuksista, vaan matemaattinen analyysi on aina tarpeen.

Projektiota valittaessa voidaan pyrkiä minimoimaan vääristymät halutulla alueella. Jos tarkoituksena on muodostaa kartta vain pienestä maapallon pinnan osasta, voidaan vääristymät minimoida sijoittamalla kuvattava alue projektioon nähden sellaiseen pisteeseen, jossa vääristymistä tapahtuu vähiten. (Galati, 2006, s. 137) Esimerkiksi joidenkin konformaalisten projektioiden tapauksessa tällainen alue sijaitsee projektion standardileveyspiirien välissä. Toisaalta kuvattava alue voi olla niin suuri, että vääristymiltä ei yksinkertaisesti voida välttyä. Projektioiden matemaattiset ominaisuudet ovat kuitenkin niin hyvin tunnettuja, että projektiosta aiheutuva virhe voidaan huomioida sovelluskohteesta riippumatta.

Bugayevskiyn ja Snyderin (1995, s. 155) mukaan useimmat valtiot käyttävät kansallisten maastokarttojen yhteydessä edellä käsitellyistä projektioista muun muassa konformaalisia ja oikeakeskipituisia tasoprojektioita, konformaalisia kartioprojektioita ja transversaalisia lieriöprojektioita. Oikeakulmaisista lieriöprojektioista yleisiä ovat erityisesti Gauss-Krüger-projektio ja UTM-projektio (*Universal Transverse Mercator*). Gauss-Krüger- ja UTM-projektioiden keskeinen ero on se, että UTM-projektiossa projektiopinta leikkaa maapallon pinnan kahdessa pisteessä, jotka ovat yhtä etäällä valitusta keskimeridiaanista. Gauss-Krüger-projektiossa projektiopinta leikkaa maapallon pinnan yhdessä pisteessä, joka vastaa valittua keskimeridiaania. (Panigrahi, 2014, s. 80–81)

Projektion valinta on siis selkeästi merkittävä yksityiskohta paikkatietoa käsiteltäessä. Galati (2006, s. 137) toteaa jopa, että "*paikkatietosovelluksen onnistuminen riippuu alkuaan projektiosta*".

Projektion valintaan liittyviä tekijöitä on useita, ja ne voidaan jaotella kolmeen ryhmään. Ensimmäisessä ryhmässä ovat projektiolla kuvattavan kohteen spatiaaliset ominaisuudet. Näitä ovat siis esimerkiksi jonkin valtion kattavaa karttaa muodostettaessa kyseisen valtion sijainti maapallolla ja sen ulkomitat. Esimerkiksi Suo-

men pohjoinen sijainti ja suhteellisen pieni koko asettavat projektiolle hyvin erilaiset vaatimukset kuin vaikkapa päiväntasaajan läheisyydessä sijaitsevan Brasilian sijainti ja mitat. Toisessa ryhmässä ovat kartan käyttötarkoitusta kuvaavat tekijät. Jos kartalla on tarkoitus havainnollistaa esimerkiksi eri alueiden pinta-alojen suhteita, on syytä valita oikeapintainen projektio. Kolmannessa ryhmässä ovat projektioiden ominaisuuksia kuvaavat tekijät, joita on käsitelty tässä kohdassa. (Bugayevskiy & Snyder, 1995, s. 235–237)

Kuten Bugayevskiy ja Snyder (1995, s. 236) mainitsevat, jo projisoitavan kohteen spatiaaliset ominaisuudet auttavat sulkemaan pois tiettyjä projektioita ja määrittelemään valittavan projektion keskeiset parametrit. Ennen lopullisen projektion valintaa voidaan siis esimerkiksi tietää, että projektion keskipisteeksi tulee valita kuvattavan alueen geometrinen keskipiste. Kuten Bugayevskiy ja Snyder esittävät, karttaprojektion valinta voidaan myös automatisoida. Automatisoinnin haasteena on kuitenkin valintaa ohjaavien tekijöiden painoarvojen objektiivinen määrittely.

4.2. Koordinaattijärjestelmät

Koordinaattijärjestelmä muodostuu datumin avulla reaalimaailmaan kiinnitetystä koordinaatistosta. Kuten kohdassa 3.2 kävi ilmi, paikkatietoa käsiteltäessä on usein tarpeen ilmaista kohteen sijainti yksikäsitteisesti. Koordinaattijärjestelmä voi olla joko globaali, alueellinen tai paikallinen. Alueellisella tarkoitetaan esimerkiksi manteretta, kun taas paikallisella tarkoitetaan esimerkiksi valtiota tai kuntaa. (Sanastokeskus TSK, 2014)

Geodeettisellä koordinaatistolla tarkoitetaan sellaista koordinaatistoa, joka ilmaisee sijainnin geodeettisen leveyden ja pituuden avulla. Tässä leveys ja pituus ovat kulmia, jotka määrittelevät horisontaalisen ja vertikaalisen sijainnin. Lisäksi tähän voi kolmiulotteisuudesta riippuen liittyä myös ellipsoidinen korkeus, joka määrittelee etäisyyden ellipsoidin keskipisteestä. Karttaprojektion koordinaatisto muodostuu yleensä kaksiulotteiseen tasoon. Se on siis tasokoordinaatisto, jossa yksikäsitteinen sijainti määritellään kahden koordinaatin avulla. Geodeettisen koordinaatiston ja kaksi- sekä kolmiulotteisen koordinaatiston välinen suhde kuvataan geodeettisella datumilla, joka määrittelee koordinaatiston origon, mittakaavan ja orientaation. (Sanastokeskus TSK, 2014)

Heywood ja muut (2006, s. 46–47) mainitsevat leveyden ja pituuden olevan äärimmäisen kattava keino maapallon pinnalla sijaitsevien kohteiden paikan määrittämiseen. Lisäksi tällaisessa koordinaattijärjestelmässä voidaan laskea esimerkiksi

pisteiden välisiä etäisyyksiä maapallon kaarevaa pintaa pitkin. Karttaesityksiin joudutaan kuitenkin käyttämään kohdassa 4.1 esiteltyjä projektioita, jolloin siirtyminen tasokoordinaatistoon tapahtuu väistämättä. Projektio kuitenkin itsessään määrittelee, miten pisteet eri koordinaatistoissa kuvautuvat toisiinsa.

Koordinaatiston määrittelemiseen tarvitaan origo sekä johtosuorat (*directrix*). Johtosuorat eivät välttämättä ole ortogonaalisia, kuten ne ovat esimerkiksi kaksiulotteisessa karteesisessa koordinaatistossa. Origo voidaan kiinnittää esimerkiksi maapallon keskipisteeseen, jos tavoitteena on kuvata esimerkiksi maapallon pinnalla sijaitsevia kiinteitä kohteita. Toisaalta tähtitieteessä koordinaatiston origo voidaan kiinnittää vaikkapa jonkin tähdistön arvioituun massakeskipisteeseen. (Panigrahi, 2014, s. 35–36)

Datumi on matemaattisesti määritelty, joten se käyttäytyy matemaattisesti ja geometrisesti täydellisesti. Datumi on siis virheetön pinta ja datumin yksittäinen piste voidaan määritellä tarkasti missä tahansa datumissa. Datumin yhdistäminen reaalimaailman pintaan tuo mukanaan väistämättä esimerkiksi mittaustarkkuudesta johtuvia virheitä, jotka ilmenevät käytettävästä koordinaattijärjestelmästä riippumatta. (Panigrahi, 2014, s. 37–38)

Paikkatietojärjestelmiä käytettäessä voidaan joutua toimimaan useamman eri koordinaattijärjestelmän kanssa samanaikaisesti, jolloin muunnokset koordinaattijärjestelmien välillä ovat tarpeen. Koordinaattijärjestelmissä tulee myös huomioida esimerkiksi se, että maapallon pinta muuttuu jatkuvasti. Mannerlaattojen liikkeet siirtävät kohteita maapallon pinnalla. Jos koordinaattijärjestelmän origo kiinnitetään esimerkiksi maapallon keskipisteeseen, pinnalla sijaitsevat kohteet liikkuvat suhteessa origoon ajan kuluessa. Tästä syystä esimerkiksi Suomen kansallinen geodeettinen datumi EUREF-FIN on kiinnitetty Euraasian mannerlaatan yhtenäiseen osaan. EUREF-FIN perustuu Euroopan terrestriseen vertausjärjestelmään 1989 (ETRS89). Terrestrisellä vertausjärjestelmällä tarkoitetaan sellaista kolmiulotteista vertausjärjestelmää, joka on sidottu jollakin tapaa maapalloon. (Julkisen hallinnon tietohallinnon neuvottelukunta, 2016a)

Suomessa maastokartoissa käytettävä ETRS-TM35FIN-koordinaattijärjestelmä käyttää EUREF-FIN-datumia. Projektiotyypiltään se on poikittainen Mercatorin projektio, jossa käytetään yhtä projektiokaistaa. Kaista ulottuu 8° keskimeridiaanista länteen ja noin 5° itään. Keskimeridiaanin sijainti on 27° itäistä pituutta. Koordinaatistolle on määritelty tarkka mittakaavakorjauksen laskentakaava. ETRS-TM35FIN-koordinaattijärjestelmällä projektiosta johtuva mittakaavavirhe on melko suuri Suomen reuna-alueilla. Mittakaavavirheen yksikkönä käytetään miljoona-

sosia, *ppm*. Koska koordinaatiston akselien yksikkö on metri, mittakaavavirheen yksikkö on millimetriä kilometrillä. Mantereella mittakaavakorjaus vaihtelee välillä $[-400ppm, +700ppm]$. Äärimmäisen tarkkoihin sovelluksiin suositellaankin käytettäväksi kapeita projektiokaistoja käyttävää ETRS-GKn -koordinaattijärjestelmää. (Julkisen hallinnon tietohallinnon neuvottelukunta, 2016b)

5 Reitinhakualgoritmit

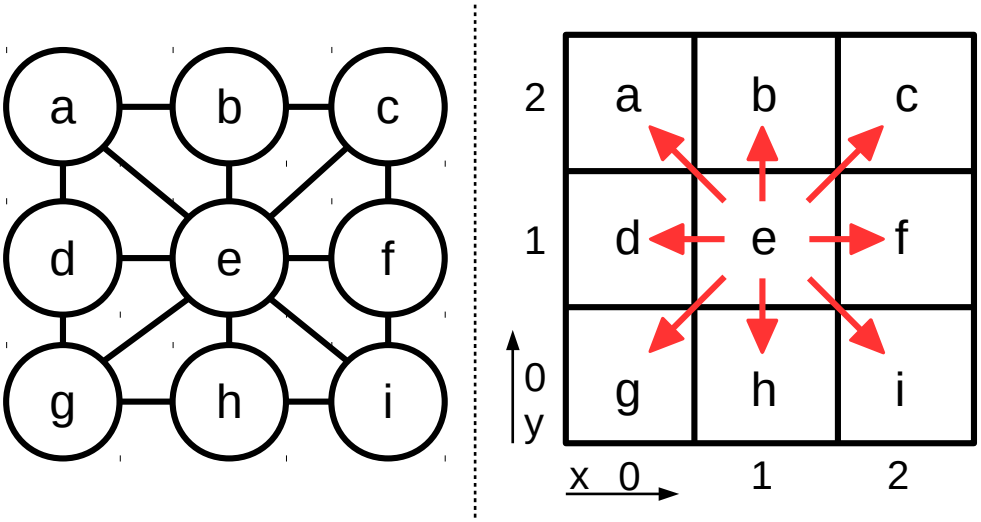
Reitinhaun ongelma tulee esille monilla eri tietotekniikan sovellusalueilla. Monet ongelmat esimerkiksi verkkoliikenteen reititykseen tai videopelihahmojen liikkumiseen liittyen ovat mallinnettavissa reitinhakuongelmina. Reitinhakualgoritmit käsittelevät tyypillisesti graafeja ja niiden tavoitteena on löytää reitti alkusolmusta yhteen tai useampaan maalisolmuun. Usein on tarpeen etsiä jollakin tapaa optimaalinen reitti. Optimaalisuus voi reitinhaun yhteydessä tarkoittaa esimerkiksi reitin pituutta. Graafissa tavoitteena voi olla löytää solmujen välille sellainen polku, joka koostuu mahdollisimman pienestä määrästä solmuja yhdistäviä kaaria. Jos kaarien painot vaihtelevat, voi reitinhakualgoritmi pyrkiä etsimään polun jonka kaarien painojen summa on mahdollisimman pieni.

Tässä luvussa esittelen reitinhakualgoritmien toimintaperiaatetta erityisesti erilaisilla ruudukoilla. Käsittelen ensin graafin ja ruudukon välisen yhteyden, jonka jälkeen esittelen kolme erilaista reitinhakualgoritmia. Tarkastelen reitinhakua erityisesti spatiaalisten esteiden näkökulmasta, sillä erilaiset esteet ja hidasteet ovat keskeinen elementti maastokartoilla reittejä haettaessa.

Graafin matemaattinen määritelmä ei ota kantaa solmujen sijaintiin avaruudessa. Graafin määritelmään $G = (V, E)$ kuuluu vain solmuista koostuva joukko V ja graafin kaaret määrittelevä kaarijoukko $E = \{(u, v) : u, v \in V\}$. Kaarien painoiksi voidaan mallintaa esimerkiksi solmujen välisiä etäisyyksiä jossakin avaruudessa. Lähtökohtaisesti kuitenkin graafi on spatiaalisesti abstrakti ja sen solmut voidaan piirtää kaksiulotteiseen tasoon mihin tahansa asetelmaan kaarijoukon E määritellessä, kuinka solmujen välillä voidaan liikkua. Solmuilla ei ole mitään kiinnitettyä sijaintia, mutta graafia tutkimalla voidaan esimerkiksi selvittää, mitkä solmut ovat vierekkäisiä eli yhden tai useamman kaaren yhdistämiä. (Ruohonen, 2013)

Graafeille suunniteltuihin reitinhakualgoritmeihin voidaan lisätä spatiaalinen ulottuvuus esimerkiksi ruudukon avulla. Ruudukko voidaan mieltää graafin erityistapauksena, jossa yksittäinen ruutu muodostaa solmun. Kuvassa 10 vasemmalla on esitetty graafi, joka koostuu solmujoukosta $\{a, b, c, d, e, f, g\}$ ja näitä yhdistävistä kaarista. Kuvassa oikealla on 3×3 ruudukko, josta graafi on muodostettu. Vasemmalla puolella esitetyn graafin solmuilla ei ole mitään tasossa määriteltyä sijaintia, eikä niihin voida viitata minkään koordinaatiston avulla. Ruudukossa taasen on määritelty koordinaatisto, joka muodostetaan x- ja y-akseleiden avulla. Esimerkiksi solmun e sijainti ruudukossa on koordinaattien avulla ilmaistuna $(1, 1)$.

Kuvan 10 graafi on muodostettu siten, että kunkin ruudun 8-naapurusto on yh-



Kuva 10. Ruudukon mallintaminen graafina reitinhakua varten.

distetty ruudusta muodostettuun solmuun kaarella. Ruudukkoon piirretyt punaiset nuolet kuvaavat ruudun e 8-naapurustoa. Voidaan ajatella, että reitinhakualgoritmi voi kulkea ruudusta e mihin tahansa kahdeksasta viereisestä ruudusta. Graafiksi mallinnettaessa tämä naapurusto ilmenee kahdeksana kaarena, jotka yhdistävät solmun e solmuihin a, b, c, d, f, g, h ja i. Reitinhakualgoritmi kykenee 8-naapuruston tapauksessa etenemään ruudukossa pysty- ja vaakasuunnassa sekä diagonaalisesti. Kuvan 10 tapauksessa voitaisiin myös estää diagonaalinen liikkuminen käyttämällä 4-naapurustoa. Tällöin solmun e viereisiä solmuja olisivat vain solmut b, d, f ja h.

Kuvassa 10 esitetyn kaltaiseen ruudukkoon voidaan siis soveltaa graafeille suunniteltuja reitinhakualgoritmeja, jos ruutujen vierekkäisyydestä voidaan johtaa jollakin tapaa kunkin ruudun naapuriruudut. Tässä luvussa käsittelen reitinhakualgoritmeja tavanomaisen neliöruudukoinnin näkökulmasta, mutta esittelen tarkemmin erilaisia ruudukointitapoja kohdassa5.4. Käytetyllä ruudukoinnilla ei ole korkealla tasolla merkitystä reitinhakualgoritmin kannalta. Ruudukointitavasta riippumatta on syntyvä hilarakenne jollakin tapaa mallinnettava graafiksi kuten kuvassa 10, jotta reitinhakualgoritmeja voidaan soveltaa.

Käytännön sovelluksissa esimerkiksi olemassa olevaa taulukkorakennetta ei ole missään tapauksessa tarpeen muuttaa graafin kaltaiseksi tietorakenteeksi. Reitinhakualgoritmit voidaan toteuttaa siten, että ne käsittelevät esimerkiksi kaksiulotteista taulukkoa graafin sijaan. Keskeistä on vain edellä mainittu naapuruston määrittely.

5.1. Dijkstran algoritmi

Yksi tunnetuimmista lyhyimmän polun hakuun kehitetyistä menetelmistä on Dijkstran algoritmi, jonka Edsger W. Dijkstra esitteli vuonna 1959. Sen tavoitteena on löytää lyhyin polku kahden solmun välillä yhdistetyssä graafissa. Lyhyimmällä polulla tarkoitetaan tässä sellaista polkua, jossa on pienin määrä kaaria. Dijkstran algoritmi etsii lähtösolmusta u alkaen lyhimpiä polkuja muihin graafin solmuihin, kunnes maalisolmu v saavutetaan. Ideana on, että jos solmu w sijaitsee solmujen u ja v välisellä lyhyimmällä polulla, tulee algoritmin löytää lyhyin polku myös solmujen u ja w välille. Dijkstran algoritmia voidaan käyttää myös minimivirittävän puun luomiseen. Tällöin hakua ei pysäytetä maalisolmun v löydyttyä, vaan haku jatkuu, kunnes kaikki graafin solmut on käsitelty ja lyhyimmät etäisyydet lähtösolmusta niihin on määritelty. (Dijkstra, 1959)

Algoritmi 1 Dijkstran algoritmi.

```
1:  function LYHYINPOLKU(V, E, u, v)
2:      Q ← V
3:      D[w] ← ∞ kaikille w ∈ V
4:      P[w] ← null kaikille w ∈ V
5:      D[u] ← 0
6:      while |Q| > 0 do
7:          w ← s : D[s] ≤ D[h] kaikille h ∈ Q
8:          if w = v then break
9:          Q ← Q − {w}
10:         for n ∈ naapurit(w, E) ∩ Q do
11:             d ← D[w] + etäisyys(w, n, E)
12:             if d < D[n] then
13:                 D[n] ← d
14:                 P[n] ← w
15:     return P, D
```

Algoritmi 1 kuvaa Dijkstran algoritmin toiminnan pseudokoodin avulla. Sen parametreina on solmujen joukko V, kaarien joukko E, lähtösolmu u sekä maalisolmu v. Tässä algoritmissa kullekin kaarelle on määritelty jokin paino, joka kuvaa kaaren yhdistämien solmujen välistä etäisyyttä. Aluksi algoritmi alustaa joukon Q vastaamaan kaikkien solmujen joukkoa V. Joukkoa Q käytetään vierailemattomien solmujen tallentamiseen. Taulukkoon D tallennetaan kutakin graafin solmua koh-

den löydetty lyhin etäisyys lähtösolmusta u. Aluksi tämä etäisyys alustetaan lähtösolmua u lukuun ottamatta kaikille solmuille äärettömäksi. Etäisyys lähtösolmusta u lähtösolmuun u on luonnollisesti 0. Taulukkoon P tallennetaan tieto kutakin solmua edeltävästä solmusta lyhyimmällä polulla. Algoritmin suorituksen alussa mitään edeltävyyssuhteita ei ole olemassa, joten kaikille solmuille tallennetaan taulukon P arvoksi tyhjä arvo, *null*.

Kuten algoritmista 1 nähdään, Dijkstran lyhyin polku käsittelee vierailemattomia solmuja, kunnes kaikissa solmuissa on vierailtu tai maalisolmu v on kohdattu. Riviltä 6 alkavassa silmukassa poimitaan käsiteltäväksi solmuksi w vierailemattomien solmujen joukosta Q sellainen solmu s, jonka etäisyys $D[s]$ on pienin kaikkien vierailemattomien solmujen joukossa Q. Jos solmu w on maalisolmu, lyhyin polku solmujen u ja v välille on löydetty ja algoritmin suoritus voidaan lopettaa. Muutoin solmu w tulkitaan vierailluksi ja poistetaan vierailemattomien solmujen joukosta Q rivillä 9. Tämän jälkeen haku laajenee käsittämään solmun w vierailemattomat naapurit. Rivillä 10 kutsuttu $naapurit(w, E)$-funktio palauttaa siis solmun w viereiset solmut graafissa. Parametrina tälle funktiolle on annettu kaarien joukko E, sillä siitä voidaan johtaa kunkin solmun naapurisolmut. Naapurisolmuista on tarpeen tarkastella vain vierailemattomia solmuja, joten naapurijoukosta otetaan leikkaus joukon Q kanssa.

Dijkstran algoritmi käsittelee solmun w vierailemattomat naapurit laskemalla kullekin naapurille n etäisyyden d algoritmin 1 rivillä 11. Etäisyys d lasketaan hyödyntämällä tietoa siitä, että $D[w]$ on toistaiseksi lyhyin löydetty polku lähtösolmusta u solmuun w. Näin ollen $D[w] + etäisyys(w, n, E)$ on pienin etäisyys, joka voidaan saavuttaa kulkemalla lähtösolmusta u solmun w kautta solmuun n. Tässä $etäisyys(w, n, E)$ on solmujen w ja n välisen kaaren painon määrittelevä funktio. Solmulle n solmun w kautta laskettu etäisyys d korvaa aiemmin löydetyn lyhyimmän etäisyyden solmulle n jos uusi etäisyys on pienempi. Tämä vertailu tapahtuu algoritmin 1 rivillä 12. Jos uusi, solmun w kautta kulkeva polku on lyhyempi, merkitään sen pituus taulukkoon D solmun n etäisyydeksi. Samoin solmun n edeltäväksi solmuksi merkitään taulukkoon P solmu w.

Kun algoritmissa 1 esitetty Dijkstran lyhyin polku saavuttaa maalisolmun v, voidaan algoritmin suoritus keskeyttää. Koska algoritmi kasvattaa hakurintamaansa lisäämällä aina lähimmän solmun voidaan todeta, että kohdattaessa maalisolmu v sitä ei voida enää saavuttaa polulla joka on lyhyempi. Tässä pseudokoodissa algoritmi palauttaa taulukot P ja D. Taulukko D sisältää kullekin solmulle lasketun etäisyyden lähtösolmusta u. Taulukosta P taas voidaan johtaa algoritmin löytämä

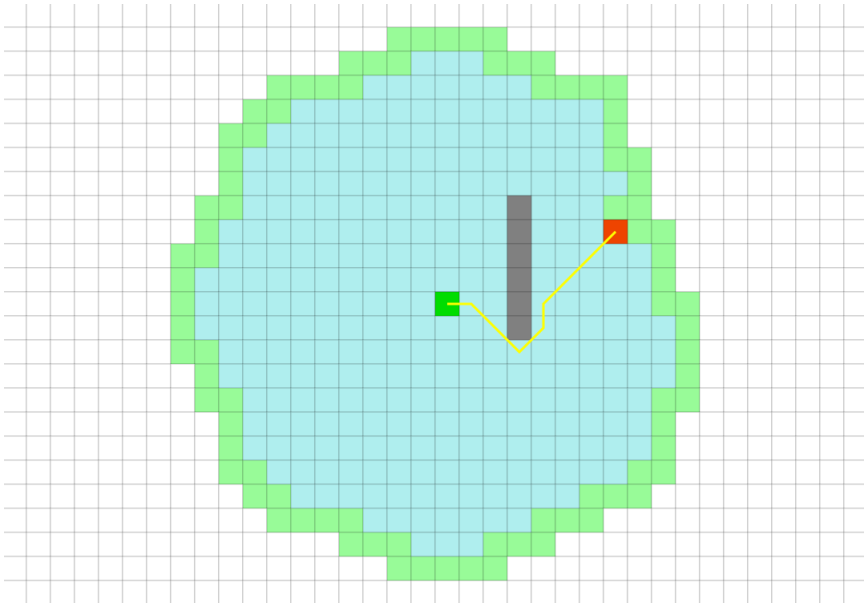
lyhyin polku, sillä siihen on merkitty kunkin solmun edeltävä solmu lyhyimmällä polulla.

Algoritmi 2 Lyhyimmän polun rakentaminen algoritmin 1 taulukosta P.

```
1: function RAKENNAPOLKU(P, u, v)
2:     Polku ← [ ]
3:     s ← v
4:     while P[s] do
5:         Polku ← [s] + Polku
6:         s ← P[s]
7:     Polku ← [u] + Polku
8:     return Polku
```

Algoritmin 2 pseudokoodi esittää, kuinka algoritmin 1 palauttamasta taulukosta P voidaan rakentaa Dijkstran algoritmin löytämä lyhyin polku. Algoritmin parametreina on taulukko P sekä lähtö- ja maalisolmut u ja v. Polku määritellään aluksi tyhjänä listana. Algoritmi kasaa polun aloittaen maalisolmusta v, joka alustetaan tarkasteltavaksi solmuksi s rivillä 3. Rivin 4 silmukka jatkaa polun rakentamista, kunnes kohdataan solmu, jolle ei ole määritelty edeltävää solmua. Kuten algoritmista 1 huomataan, lähtösolmulle u jää edeltäväksi solmuksi algoritmin alussa alustettu tyhjä arvo, *null*. Silmukan sisällä jokainen kohdattu solmu s lisätään polun alkuun, sillä algoritmi etenee polkua maalista alkua kohti. Taulukosta P valitaan aina seuraavaksi solmuksi kulloinkin tarkasteltua solmua edeltävä solmu $P[s]$. Lopuksi rivillä 7 lisätään vielä alkusolmu u polun ensimmäiseksi solmuksi, sillä silmukassa sitä ei käsitellä $P[u]$:n ollessa määrittelemätön.

Dijkstran algoritmi toimii siis laajentamalla eräänlaista hakurintamaa joka suuntaan, kunnes maalisolmu löytyy. Hakurintaman laajentamissääntö takaa sen, että löytyvä polku on lyhyin mahdollinen. Kuvassa 11 on esitetty ruudukko, johon on sovellettu Dijkstran algoritmia. Lähtöruutu on merkitty vihreällä ja maaliruutu punaisella. Algoritmin löytämä reitti on merkitty kuvaan keltaisella viivalla, joka kulkee lyhyimmän polun määrittelemien ruutujen keskipisteiden kautta. Kuten kuvasta huomataan, tässä algoritmin suorituksessa on käytetty ruuduille 8-naapurustoa. Ruudukkoon on lisäksi määritelty este harmaina ruutuina. Nämä harmaat ruudut eivät siis ole osa kuljettavissa olevaa ruudukkoa. Dijkstran algoritmin näkökulmasta ne eivät ole minkään ruudun naapuriruutuja, joten hakurintama ei voi edetä niihin. Maastokartoilla reitinhakua toteutettaessa esimerkiksi vesistöt voitaisiin mieltää tällaisiksi esteiksi.



Kuva 11. Dijkstran algoritmin eteneminen ja löydetty reitti ruudukossa. (Xu, 2017)

Kuvassa 11 Dijkstran algoritmin läpikäymät ruudut on merkitty turkoosilla värillä. Turkoosia aluetta ympäröi vaaleanvihreä reunus, joka kuvaa algoritmin hakurintamaa. Kuvasta nähdään, kuinka Dijkstran algoritmi etenee lähtöruudusta joka suuntaan. Esteen suuntaan edennyt hakurintaman osa jää hieman jälkeen ruudukossa vasemmalle edenneestä hakurintamasta. Este pidentää sen oikealle puolelle jääviin ruutuihin kulkevia polkuja, sillä sen läpi ei voida edetä. Kuva 11 havainnollistaa hyvin, kuinka Dijkstran algoritmi ei hyödynnä spatiaalista ulottuvuutta vaan laajentaa hakurintamaa maaliruudun sijainnista riippumatta jokaiseen suuntaan. Tavanomaisessa graafissa nämä suunnat ovat merkityksettömiä. Vasta ruudukkoesityksessä ne voidaan huomioida. Myöhemmin tässä luvussa esittelemäni hakualgoritmit hyödyntävät ruudukoiden luomaa spatiaalista ulottuvuutta.

Kuvassa 11 esitetty este on luonteeltaan binäärinen. Jokainen ruutu on joko osa estettä tai kuljettavissa. Ruudukkoon voitaisiin myös mallintaa sellaisia esteitä, joiden kuljettavuus on tyhjää ruutua heikompi, mutta ne ovat kuitenkin kuljettavissa. Tällaiseen esteeseen voidaan edetä toisin kuin binäärisesti määriteltyyn esteeseen. Ruudun kuljettavuus välittyy Dijkstran reitinhakualgoritmille vierekkäisten ruutujen välisenä etäisyytenä, jota vertaillaan esimerkiksi algoritmin 1 rivillä 11. Tällöin algoritmin hakurintama siis laajenee heikosti kuljettavissa olevien ruutujen läpi hitaammin kuin hyvin kuljettavissa olevien ruutujen. Kuljettavuus voi olla esimerkiksi reaaliluku välillä $[1, 10]$. Tällöin etäisyys esteettömästä ruudusta esteettömään naapuriin on 1, ja etäisyys esteettömästä ruudusta huonommin kuljettavaan naapuri-

ruutuun on välillä $]1, 10]$.

Dijkstran algoritmin aikakompleksisuus on $O(|V|^2)$, missä $|V|$ on solmujen lukumäärä. Algoritmia voidaan kuitenkin tehostaa huomattavasti käyttämällä prioriteettijonoa, kuten Fredman ja Tarjan (1987) osoittavat. Jos käytetään heidän kuvailemaansa Fibonaccin kekoa, algoritmin aikakompleksisuus on $O(|V|log|V|+|E|)$, missä $|V|$ on solmujen lukumäärä ja $|E|$ kaarien lukumäärä. Prioriteettijonon käyttäminen solmujen valintaan hakurintamasta on ilmeinen keino algoritmin tehostamiseksi, ja prioriteettijonoja käytetään muissakin hakualgoritmeissa laajalti. Algoritmissa 1 prioriteettijonoa käytettäisiin siis rivillä 7, prioriteettiavaimena etäisyystaulukon $D[s]$ arvo kullekin jonoon asetetulle solmulle s. Naapurustoa käsiteltäessä naapurisolmut lisättäisiin tähän prioriteettijonoon. Mielenkiintoisena yksityiskohtana käytettävän prioriteettijonon tulee toteuttaa jonoon jo asetettujen olioiden prioriteettien muuttaminen. Jos jonkin solmun kautta löydetty polku onkin lyhyempi kuin aiemmin löydetty polku, prioriteettijonossa jo olevan solmun etäisyys tulee päivittää.

Dijkstran algoritmi takaa lyhyimmän polun löytymisen graafissa, jonka kaarien painot ovat positiivisia. Negatiivisten painojen tapauksessahan löytyvän lyhyimmän polun pituutta voidaan pienentää äärettömästi kiertämällä uudestaan ja uudestaan silmukkaa, jonka kokonaispaino on negatiivinen. (Dijkstra, 1959)

5.2. A*

Kuten edellisessä kohdassa mainitsin, Dijkstran (1959) algoritmi voi huomioida ruudukossa vaihtelevan edettävyyden laajentamalla hakurintamaa ensisijaisesti niihin suuntiin, jotka ovat helposti edettävissä. Algoritmi ei kuitenkaan huomioi spatiaalista ulottuvuutta, eli se ei etene ensisijaisesti siihen suuntaan, jossa maalisolmun tiedetään olevan. Hartin, Nilssonin ja Raphaelin (1968) esittelemä A*-algoritmi lisää perinteiseen reitinhakuun ongelman kohdealueesta muodostetun heuristiikan, joka ohjaa hakua kohti maalia. Hart ja muut kutsuvat algoritmia hyväksyttäväksi, jos se takaa lyhyimmän polun löytymisen missä tahansa graafissa. He osoittavat, kuinka A*-algoritmi kykenee laajentamaan hakua mahdollisimman pieneen määrään solmuja ja toimimaan silti hyväksyttävästi.

A*-algoritmin keskeisenä ajatuksena on käyttää erityistä arviointifunktiota $\hat{f}(n)$, jonka arvo voidaan laskea mille tahansa solmulle n. Reitinhaku hyödyntää tätä arviointifunktiota seuraavan tutkittavan solmun valitsemiseen. Hakurintaman solmuista valitaan siis se, jolle arviointifunktion $\hat{f}(n)$ arvo on pienin. Muutoin algoritmi

toimii hyvin samankaltaisesti kuin Dijkstran algoritmi. (Hart ja muut, 1968)

A*-algoritmissa Hart ja muut (1968) määrittelevät arviointifunktion $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$. Tässä $\hat{g}(n)$ on toistaiseksi löydetty lyhin polku lähtösolmusta u solmuun n ja $\hat{h}(n)$ on heuristinen arvio solmusta n maalisolmuun v kulkevan optimaalisen polun pituudesta. A*-algoritmissa tämä arviointifunktio $\hat{f}(n)$ pyrkii mallintamaan mahdollisimman tarkasti funktion $f(n) = g(n) + h(n)$ arvoa kullekin solmulle. Funktion $f(n)$ arvo on lähtösolmusta u optimaalisen polun pituus solmun n kautta maalisolmuun v. Tällöin funktion $g(n)$ arvo vastaa optimaalista polkua lähtösolmusta u solmuun n. Vastaavasti $h(n)$ on solmun n ja maalisolmun v välisen optimaalisen polun todellinen pituus. A*-algoritmin määrittely ei ota kantaa siihen, mitä heuristiikkaa käytetään.

Algoritmi 3 A*-algoritmi.

```
1:  function LYHYINPOLKU(V, E, u, v)
2:      O ← {u}
3:      P[w] ← null kaikille w ∈ V
4:      G[w] ← ∞ kaikille w ∈ V
5:      F[w] ← ∞ kaikille w ∈ V
6:      G[u] ← 0
7:      F[u] ← heuristiikka(u, v)
8:      while |O| > 0 do
9:          w ← s : F[s] ≤ F[h] kaikille h ∈ O
10:         if w = v then break
11:         O ← O − {w}
12:         for n ∈ naapurit(w, E) do
13:             g ← G[w] + kustannus(w, n)
14:             if G[n] > g then
15:                 G[n] ← g
16:                 F[n] ← g + heuristiikka(n, v)
17:                 P[n] ← w
18:                 O ← O + {n}
19:     return P
```

Algoritmi 3 kuvaa A*-algoritmin toiminnan pseudokoodin avulla. Funktion parametrit ovat samat kuin algoritmissa 1. Aluksi avoimien solmujen joukoksi O alustetaan pelkästä lähtösolmusta u koostuva joukko. Taulukkoon P algoritmi kasaa Dijkstran algoritmin kaltaisesti tiedon kutakin solmua edeltävästä solmusta lyhyim-

mällä polulla. Taulukko G sisältää toistaiseksi löydetyn lyhimmän polun pituuden kullekin solmulle lähtösolmusta alkaen. Taulukko F vastaa algoritmin määrittelyssä arviointifunktion $\hat{f}(n)$ arvoa kullekin solmulle. Algoritmin edetessä taulukon F arvot muodostuvat siis kullekin solmulle taulukon G ja käytetyn heuristiikan laskemien arvojen perusteella.

Kuten algoritmista 3 käy ilmi, A* lopettaa solmujen käsittelyn välittömästi kohdattaessa maalisolmu v. Jos maalisolmua ei kohdata, solmujen käsittely loppuu avoimien solmujen joukon O tyhjentyessä. Rivillä 9 valitaan käsittelemättömistä solmuista seuraavaksi se, jonka arvo F-taulukossa on pienin. Tämä vastaa Hartin ja muiden (1968) määrittelemän arviointifunktion $\hat{f}(n)$ arvoa kyseiselle solmulle. Algoritmin tehostamiseksi tässä käytetään tyypillisesti prioriteettijonoa.

A*-algoritmi käsittelee algoritmin 3 rivillä 12 solmun w naapurisolmut laskien kullekin näistä lyhyimmän polun pituuden solmun w kautta kuljettuna. Tämän polun pituus g muodostuu solmuun w löytyneestä lyhyimmästä polusta sekä erityisen *kustannus*-funktion arvosta solmuille w ja n. Tavanomaisessa graafissa tämä kustannus voisi olla solmujen w ja n välisen kaaren paino. Ruudukoissa tämä kustannus voidaan laskea esimerkiksi kyseisen naapuriruudun keskipisteen etäisyytenä ruudun w keskipisteestä. Lisäksi tässä kustannuksessa voidaan huomioida kyseisen ruudun kuljettavuus, aivan kuten Dijkstran algoritmissa.

Jos algoritmin 3 laskema polun pituus g on pienempi kuin solmulle n toistaiseksi löydetty lyhyin polku, korvaa g taulukossa G olleen arvon. Nyt A*-algoritmin arviointifunktion $\hat{f}(n)$ arvo voidaan laskea solmulle n laskemalla yhteen toistaiseksi löydetyn lyhyimmän polun pituus g, sekä käytetyn *heuristiikka*-funktion arvo solmun n ja maalisolmun v välille. Algoritmissa *heuristiikka*-funktion kutsu vastaa siis laskentaa funktion $\hat{h}(n)$ arvolle. Taulukkoon F solmulle n tallennettava arvo toimii solmun n prioriteettina, sillä algoritmin rivillä 9 avoimien solmujen joukosta valitaan se, jolla on taulukossa F pienin arvo. Rivillä 16 laskettavasta summasta huomataan, että A*-algoritmi laajentaa reitinhakua ensisijaisesti niihin solmuihin, joihin päästään mahdollisimman pienellä kustannuksella ja jotka ovat mahdollisimman lähellä maalisolmua. Tämä heuristiikka tekee A*-algoritmista tehokkaan.

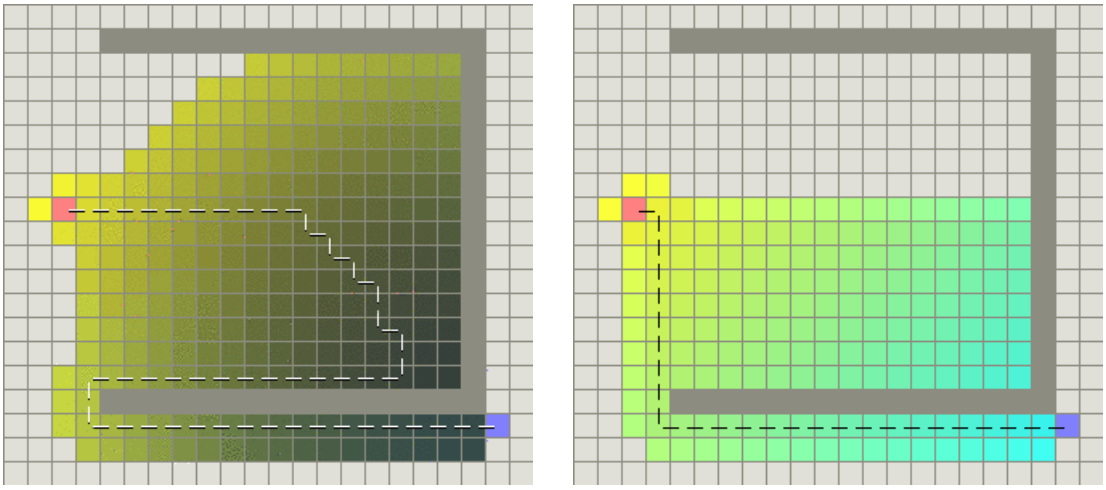
Reitinhaussa A*-algoritmiin soveltuvat heuristiikat voidaan johtaa spatiaalisesta ulottuvuudesta. Jos mitään ongelman kohdealueesta johdettua tietoa ei käytetä, heuristinen arviointifunktio on $\hat{h}(n) = 0$ kaikille solmuille. Tällöin A*-algoritmin reitinhaku etenee kaikkiin suuntiin kaksiulotteisessa tasossa ja vastaa käytännössä Dijkstran algoritmia. (Hart ja muut, 1968)

Kuten Hart ja muut (1968) toteavat, euklidisessa avaruudessa voidaan käyttää

esimerkiksi euklidista etäisyyttä heuristiikkana. Kartalle hahmotettuna euklidinen etäisyys on kahden pisteen välille piirretyn suoran viivan pituus. Tällöin funktion $\hat{h}(n)$ arvo jollekin kaksiulotteisessa tasossa koordinaateissa (x_1, y_1) sijaitsevalle solmulle n olisi sen euklidinen etäisyys maalisolmusta v koordinaateissa (x_2, y_2), eli $\hat{h}(n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Olennaista on, että käytetyn heuristiikan arvo mille tahansa solmulle on pienempi tai yhtä suuri kuin todellisen lyhyimmän polun pituus solmusta maalisolmuun. Hart ja muut (1968) käyttävät esimerkkinä toista heuristiikkaa: $\hat{h}(n) = ((|x_1 - x_2|) + (|y_1 - y_2|))/2$. Tällaisella heuristiikalla vältytään laskemasta neliöjuuria, jotka voivat joillakin alustoilla olla laskennallisesti raskaita operaatioita. Lisäksi kaksi neliöönkorotusta jää pois, mikä edelleen nopeuttaa laskentaa. Koska tällä heuristiikalla jokaiselle solmulle laskettu arvo on pienempi kuin kyseisen solmun euklidinen etäisyys maalisolmusta, voidaan osoittaa, että A*-algoritmi toimii silti hyväksyttävästi. Euklidiseen etäisyyteen verrattuna tämä neliöjuureton heuristiikka kuitenkin johtaa siihen, että A*-algoritmi käy läpi enemmän solmuja lyhyimmän polun löytämiseksi.

Kaksiulotteisella kartalla käytettävän heuristiikan tarkkuus vaikuttaa siis merkittävästi siihen, kuinka paljon A*-algoritmi joutuu käymään läpi solmuja. Patel (2017a) toteaa heuristiikan valinnan olevan tasapainottelua algoritmin nopeuden ja tarkkuuden välillä. Kuten Hartin ja muiden (1968) matemaattisesta todistuksesta käy ilmi, heuristiikan arvioidessa polun pituutta äärimmäisen tarkasti A*-algoritmi etenee täsmällisesti lyhyintä polkua pitkin suoraan maalisolmuun. Jos heuristiikka palauttaa joillekin solmuille suuremman etäisyyden kuin todellinen polun pituus, algoritmi ei toimi enää hyväksyttävästi, mutta saattaa löytää polun nopeammin. Toisaalta mitä pienempi heuristiikan arvioima polku on todelliseen polun pituuteen nähden, sitä hitaammin algoritmi toimii. Näissä tapauksissa algoritmi toimii hyväksyttävästi, ja löytyvä polku on lyhyin mahdollinen.

Patel (2017a) huomauttaa, että jos arviointifunktion heuristisen osan $\hat{h}(n)$ arvot ovat äärimmäisen suuria toistaiseksi löydetyn lyhyimmän polun $\hat{g}(n)$ arvoihin nähden, A*-algoritmin toiminta alkaa muistuttaa ahnetta paras ensin -hakua. Tämä on selvästi nähtävissä esitetystä pseudokoodista algoritmissa 3. Jos rivillä 16 asetettava arviointifunktion summa koostuu vain *heuristiikka*-funktion arvosta ja käytetty heuristiikka on euklidinen etäisyys, reitinhaku etenee aina mahdollisimman suoraan kohti maalia ahneesti ja tehdyt päätökset polusta jäävät lopullisiksi. Kuvassa 12 vasemmalla on esitetty, kuinka A*-algoritmi käyttäytyy heuristiikkana käytetyn Manhattan-etäisyyden dominoidessa arviointifunktiota. Kuvan oikealla puoliskolla esitetty reitti vastaa tilannetta, jossa arviointifunktion $\hat{h}(n)$ ja $\hat{g}(n)$ osien



Kuva 12. Vasemmalla A*-algoritmin pelkällä heuristiikalla löytämä reitti, oikealla tasapainoisella arviointifunktiolla löydetty reitti. (Patel, 2017a)

välinen suhde on tasapainoisempi. Kuvan 12 tapauksissa diagonaalinen liike ruudukossa on estetty, joten reitinhaku voi edetä ruudukossa vain vasemmalle, oikealle, ylös tai alas.

Patel (2017a) ehdottaa, että arviointifunktioon voidaan lisätä jokin kerroin $\alpha \in [0, 1]$ reitinhaun laskennan mukauttamiseksi ruudukoilla, joissa ruutujen kuljettavuus vaihtelee. Kerroin lisätään arviointifunktioon seuraavasti: $\hat{f}(n) = (1 + \alpha \times (\hat{g}(n) - 1) + \hat{h}(n)$. Nyt jos $\alpha = 1$, arviointifunktio käyttää alkuperäistä funktiota $\hat{g}(n)$ sellaisenaan. Kertoimen arvon ollessa $\alpha = 0$ ruudukon kuljettavuutta ei huomioida ja reitinhaku etenee heuristiikkaa hyödyntäen suoraan kohti maalia. Koska kerroin α voidaan asettaa välille $[0, 1]$, sillä voidaan hallita sitä, kuinka aggressiivisesti reitinhaku yrittää etsiä reittiä hankalasti kuljettavien alueiden ympäri. Kertoimen pienentyessä algoritmi siis tuottaa mahdollisesti reittejä, jotka eivät ole lyhyimpiä mahdollisia, mutta niiden laskenta-aika on merkittävästi pienempi.

Patelin (2017a) mukaan ruudukoissa käytettävät heuristiikat ovat hyvin tunnettuja ja käytettävän heuristiikan valinta on yksinkertaista. Neliön muotoisilla ruuduilla tulisi käyttää diagonaalista etäisyyttä L_∞, jos diagonaalinen liikkuminen ruudukossa on sallittu. Jos liikkuminen ruudukossa on rajoitettu vasemmalle, oikealle, alas ja ylös, tulee käyttää Manhattan-etäisyyttä L_1. Manhattan-etäisyys pisteiden $a = (x_1, y_1)$ ja $b = (x_2, y_2)$ välillä on $D \times (|x_1 - x_2| + |y_1 - y_2|)$. Tässä D on pienin kustannus ruudukossa kahden ruudun välillä. Esteettömässä ruudukossa liikuttaessa kunkin ruudun kustannus on D. Tällaisessa tilanteessa jokaisen askelen kohti maalia tulisi pienentää arviointifunktion heuristista osuutta D:n verran, ja toisaalta kas-

vattaa solmuun mitattua etäisyyttä D:n verran. Tällöin arviointifunktion osien $\hat{g}(n)$ ja $\hat{h}(n)$ voidaan katsoa olevan tasapainossa, eli niiden mittakaava on sama.

Diagonaalinen etäisyys L_∞ lasketaan erottelemalla viistoon liikkumisen kustannukset muista suunnista. Tällöin edellä mainittujen pisteiden a ja b välinen etäisyys on $D_1 \times (d_x + d_y) + (D_2 - 2 \times D) \times min\{d_x, d_y\}$, missä d_x on $|x_1 - x_2|$ ja d_y on $|y_1 - y_2|$. Tässä D_1 on ylös, alas, vasemmalle ja oikealle liikkumisen kustannus. Vastaavasti D_2 on viistoon liikkumisen kustannus. Jos esimerkiksi neliön muotoisilla ruuduilla asetetaan $D_1 = 1$, niin $D_2 = \sqrt{2}$.

Edellä mainituista heuristiikoista voidaan toki poiketa, jos reitinhaun sovellusalue asettaa esimerkiksi laskennallisia erityisvaatimuksia. Kuten esimerkiksi Dechter ja Pearl (1985) toteavat, A*-algoritmin käyttäminen hyväksyttävästi rajoittaa usein algoritmin tehokkuutta merkittävästi. He esittävät, kuinka algoritmin käyttämää arviointifunktiota voidaan muokata laskennallisen tehokkuuden parantamiseksi siten, että löytyvät reitit ovat mahdollisimman lähellä optimaalista.

A*-algoritmista huomataan, että se tarjoaa huomattavaa joustavuutta reitinhaun tarpeisiin. Arviointifunktion muokkaaminen ja α-kertoimen lisääminen vaikuttavat algoritmin toimintanopeuteen ja toisaalta löytyvien reittien optimaalisuuteen. Tämä lisää ohjelmoijan vastuuta merkittävästi, sillä sopivien parametrien ja arviointifunktioiden rakenteen löytäminen vaatii selvitystyötä ja testausta. Käytettävien parametrien valintaa voidaan joutua tekemään dynaamisesti, jos esimerkiksi lähtö- ja maalisolmujen etäisyydet vaihtelevat paljon. Maastokartalla reitinhaku voi kohdistua valtavan suureen ruudukkoon, jossa ruutujen kustannuksissa on suuria vaihteluita. Tällöin hyväksyttävästi toimiva A*-algoritmi joutuu käsittelemään tarpeettoman suuren määrän ruutuja etsiessään reittiä esimerkiksi vesistöjen tai korkeiden maastonkohtien ympäri.

Toisaalta A*-algoritmin muokattavuus voidaan avata reitinhaun loppukäyttäjälle. Tällöin loppukäyttäjät voivat erilaisten abstraktioiden kautta vaikuttaa reitinhaun toimintaan. Yksi tällainen abstraktio voisi olla esimerkiksi liukusäädin joka määrittelee reitinhaun nopeuden ja löytyvän reitin pituuden välisen tasapainon. Liukusäätimen toisessa ääripäässä algoritmi toimii hyväksyttävästi mutta reitinhaku saattaa kestää pidempään. Toisessa ääripäässä taasen algoritmin löytämät reitit eivät ole enää optimaalisia, mutta reitinhaku tutkii pienemmän määrän ruutuja ja saavuttaa maalin nopeammin. Tällainen liukusäädin on kohtalaisen intuitiivinen käyttää, mutta esimerkiksi Dectherin ja Pearlin (1985) esittelemän A**-variantin käyttämän arviointifunktion ominaisuuksien intuitiivinen kuvaus voi osoittautua hankalaksi.

5.3. Iteratiivisesti syvenevä A*

Kuten kohdasta 5.2 kävi ilmi, A*-algoritmi tallentaa hakurintamasta löytyviä solmuja myöhempää käsittelyä varten johonkin tietorakenteeseen, esimerkiksi prioriteettijonoon. Tämän tietorakenteen kasvu haun edetessä voi tuottaa ongelmia muistinkäytön suhteen. Korfin (1985) kehittämä iteratiivisesti syvenevä A*-algoritmi (*iterative deepening A*, IDA**) toteuttaa reitinhaun syvyys-ensin -menetelmällä, jolloin tarve muistille pienenee huomattavasti. Iteratiivisesti syvenevä A* perustuu Korfin esittelemään yleiseen iteratiivisesti syvenevään syvyyshakuun. Tällaisessa syvyyshaussa tehdään useita iteraatioita, joissa haun etenemissyvyyttä kasvatetaan jatkuvasti. Syvyydellä tarkoitetaan tässä yhteydessä etäisyyttä lähtösolmusta.

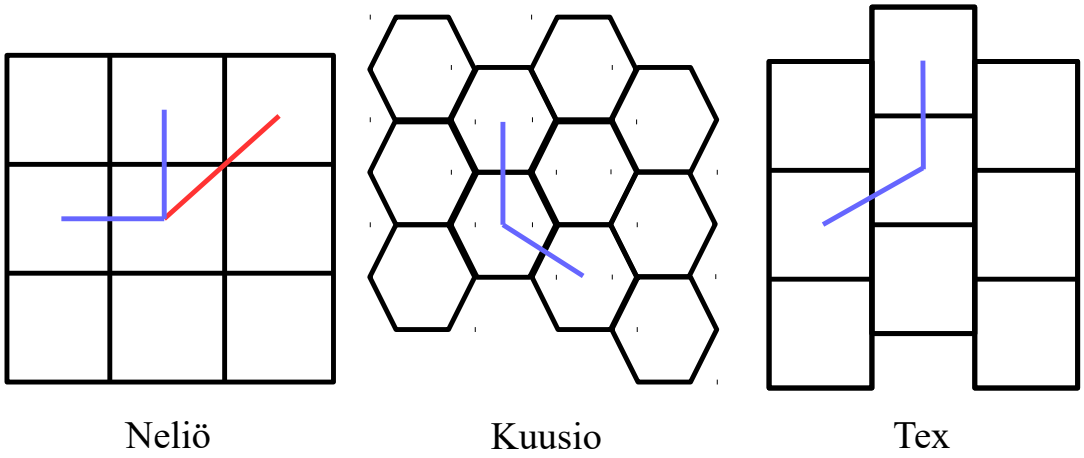
Iteratiivisesti syvenevässä A*-algoritmissa yhdistetään iteratiivisesti syvenevä syvyyshaku A*-algoritmin arviointifunktioon $\hat{f}(n)$. Iteraatioiden välillä rajoitetaan etenemissyvyyden sijaan arviointifunktion palauttamia arvoja. Reitinhaku vertailee edetessään solmuille laskettua arviointifunktion arvoa tiettyyn kynnysarvoon. Mikäli kynnysarvo ylittyy jossakin solmussa, syvyyshaku kyseiseen haaraan keskeytetään. Algoritmin alussa kynnysarvoksi määritellään arviointifunktion arvo lähtösolmulle. Jokaisen iteraation jälkeen kynnysarvoksi valitaan edellisen iteraation pienin kynnysarvon ylittänyt arvo. Iteratiivisesti syvenevä A*-algoritmi toimii A*-algoritmin lailla hyväksyttävästi, jos käytetty heuristiikka ei koskaan yliarvioi etäisyyttä maalisolmuun. (Korf, 1985)

Korf (1985) osoittaa, että IDA*-algoritmi löytää hyväksyttävää heuristiikkaa käytettäessä optimaalisen polun ja käyttää huomattavasti vähemmän muistia kuin A*-algoritmi. Lisäksi IDA*-algoritmin käsittelemien solmujen määrä on likimäärin sama kuin A*-algoritmin. IDA*-algoritmi on myös yksinkertaisempi toteuttaa, sillä avoimista ja suljetuista solmuista ei ole tarpeen pitää kirjaa. Syvyyshaku voidaan toteuttaa rekursiivisesti ja iteraatiot voidaan käsitellä silmukassa. Algoritmin muistinkäyttö on lineaarinen löytyvään lyhyimpään polkuun nähden. Koska algoritmi ei edetessään pidä kirjaa käsitellyistä solmuista, se voi päätyä käsittelemään samoja solmuja useampaan kertaan.

5.4. Kartan ruudukointi

Edellä käsiteltyjen reitinhakualgoritmien yhteydessä käytetyt ruudukot muodostuivat yksinomaan neliön muotoisista ruuduista, jotka on sijoitettu säännölliseen asetelmaan riveittäin ja sarakkeittain. Kutakin ruutua ympäröi siis kahdeksan ruutua, ja jokainen neliö jakaa täsmälleen yhden sivun neljän naapurinsa kanssa. Viistossa

olevien neljän naapurin kanssa kukin ruutu jakaa vain yhden pisteen. Ruudukointitapoja on kuitenkin monia, ja erityisesti kohdassa 5.3 esittelemäni iteratiivisesti syvenevän A*-algoritmin toiminnan kannalta käytetyllä ruudukoinnilla on suuri merkitys.



Kuva 13. Ruudukointitapoja ja ruutujen väliset etäisyydet niissä. (Bjornsson ja muut, 2003, mukaillen)

Kuvassa 13 on esitetty kolme erilaista ruudukointitapaa. Vasemmalla on edellisissä kohdissa käytetty neliöruudukko. Keskimmäisestä ruudusta piirretyillä sinisillä viivoilla on kuvattu etäisyyttä ruudun keskipisteestä ruudun yläpuolella ja vasemmalla olevien naapuriruutujen keskipisteisiin. Jos tämän sinisen viivan pituudeksi määritellään yksi, on viistoon edettäessä etäisyys ruutujen keskipisteiden välillä $\sqrt{1^2 + 1^2} = \sqrt{2} \approx 1.4142$. Tästä huomataan, että neliöruudukossa etäisyydet ruutujen välillä vaihtelevat, jos viistoon liikkuminen on sallittu. Tämä tulee huomioida määriteltäessä ruudukossa liikkumisen kustannuksia reitinhakualgoritmeja varten. Jos tätä eriävää etäisyyttä ei huomioida, reitinhaku voi viistoon etenemällä liikkua pidemmän matkan ruudukon peittämässä spatiaalisessa avaruudessa ilman ylimääräistä kustannusta. Mikäli viistoon liikkumista ei ole sallittu, etäisyys kaikkiin ruudun naapureihin on sama. (Bjornsson ja muut, 2003)

Kuvan 13 keskimmäinen kuusioruudukko (*hexagon grid*) koostuu säännöllisistä kuusikulmioista. Säännöllisen kuusikulmion kaikki sivut ovat yhtä pitkiä ja niiden välinen kulma on 120 astetta. Kuvaan on piirretty sinisellä viivalla etäisyydet yhden ruudun keskipisteestä sen yläpuolella ja viistossa alaoikealla oleviin naapureihin. Topologisesti kullakin ruudulla on tässä esityksessä kuusi naapuriruutua, yksi kullakin ruudun sivulla. Etäisyys kaikkiin ruudun naapureiden keskipisteisiin on sama. Kuten

Bjornsson ja muut (2003) toteavat, kuusioruudukkoa ei voida toteuttaa käytännössä yhtä tehokkaasti kuin neliöruudukkoa. Esimerkiksi Patel (2017b) käy perusteellisesti läpi kuusioruudukoiden geometriaa, niihin soveltuvia koordinaatistoja ja muita toteutusyksityiskohtia.

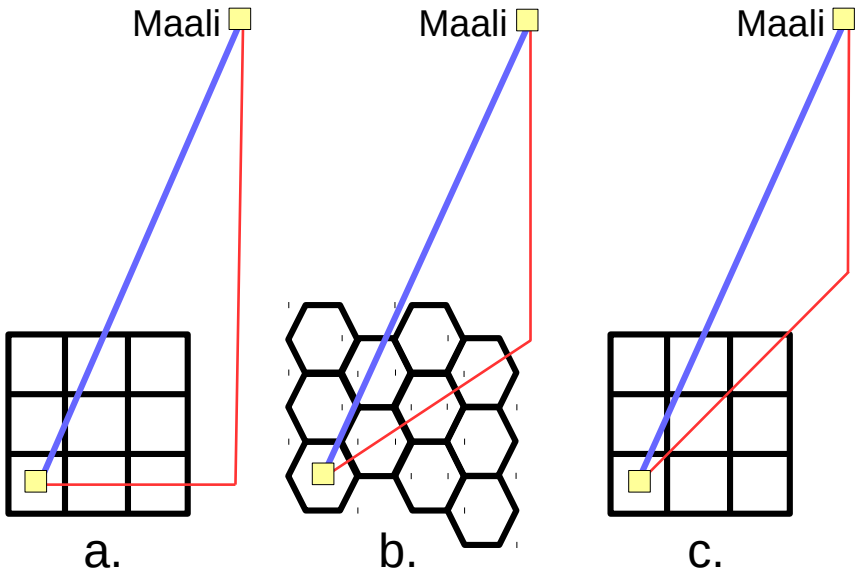
Yapin (2002a) esittelemä tex-ruudukko on kuvassa 13 oikealla. Sen ruudut ovat neliöitä, mutta topologisesti se vastaa kuusioruudukkoa. Ruudukon sarakkeista joka toinen on nostettu puolet ruudun korkeudesta ylöspäin. Koska tex-ruudukon ruudut ovat neliöitä, niiden geometria on huomattavasti yksinkertaisempi kuin kuusioiden. Kullakin ruudulla on tässä ruudukointitavassa kuusi naapuriruutua. Yksittäinen ruutu jakaa kunkin naapurinsa kanssa joko yhden kokonaisen sivun tai puolet sivusta. Kuvaan on piirretty sinisillä viivoilla etäisyydet yhden ruudun keskipisteestä sen yläpuolella ja alavasemmalla oleviin naapureihin. Kuten kuusioruudukossakin, tex-ruudukossa etäisyys ruudusta kaikkiin sen naapureihin on sama.

A*-algoritmin hakukompleksisuus on yksikkökustanteisessa ruudukossa $O(D^2)$. Tässä D on algoritmin löytämän polun syvyys, eli etäisyys ruudukossa lähtösolmusta. Merkittävää on, että A*-algoritmin hakukompleksisuus ei riipu käytetystä ruudukointitavasta. Toisaalta IDA*-algoritmin tapauksessa ruudukoinnilla on kuitenkin suuri vaikutus. (Bjornsson ja muut, 2003) Kuten Korf, Reid ja Edelkamp (2001) osoittavat, IDA*-algoritmin kompleksisuus on $O(b^D)$, jos heuristiikan vaikutusta ei huomioida. Tässä b on käytetyn ruudukoinnin keskimääräinen haarautumiskerroin. Haarautumiskertoimella tarkoitetaan sitä, kuinka montaa naapurisolmua algoritmin tarvitsee käsitellä saapuessaan johonkin solmuun. Optimaalisella polulla ei koskaan tarvitse mennä takaisin ruutuun josta on jo tultu. Jos esimerkiksi neliöruudukossa käytetään 4-naapurustoa, haarautumiskerroin b on 3 ja kompleksisuus syvyydessä D sijaitsevalle ratkaisulle on täten $O(3^D)$. (Yap, 2002a)

Kuten Yap (2002a) havainnollistaa, kuusioruudukolla haarautumiskerroin on kolme kuten neliöruudukoidulla 4-naapurustollakin. Vaikka kustakin kuusiosta voidaan liikkua kuuteen eri suuntaan, voidaan kolme suunnista poissulkea. Liikuttaessa esimerkiksi jostakin kuusioruudusta a suoraan ylöspäin ruutuun b, voidaan ruudun b alaviistossa vasemmalla ja oikealla olevat naapurit sulkea pois edellisen kuusion a lisäksi. Jos ruudun b alaviistossa vasemmalla ja oikealla olevat kuusiot olisivat osa optimaalista reittiä, algoritmi olisi siirtynyt niihin jo ruudusta a. Tämä toki pitää paikkansa vain silloin, kun kaikkien ruutujen väliset kustannukset ovat positiivisia.

Koska kuusioiden ja neliöiden geometriset ominaisuudet ovat erilaisia, polkujen pituuksia vertaillessa oletetaan sekä neliöiden että kuusioiden pinta-alat samoiksi. Tämä tasavertaistaa muodot, sillä niitä voidaan katsoa tarvittavan keskimäärin

sama määrä tietyn alueen pinnan peittämiseksi.



Kuva 14. Polkujen teoreettiset minimipituudet erilaisilla ruudukoilla. (Bjornsson ja muut, 2003, mukaillen)

Haarautumiskertoimen lisäksi toinen merkittävä ruudukointitapojen välinen ero on löytyvän polun pituus. Yapin (2002b) mukaan kuusioruudukossa määritelty polku on keskimäärin lyhyempi kuin 4-naapurustoisessa neliöruudukossa. Kuvassa 14 on havainnollistettu, kuinka polkujen teoreettiset minimipituudet vaihtelevat yksikkökustanteisilla ruudukoilla. Kuvan vasemmassa laidassa kohdassa a. on esitetty, kuinka neliöruudukoidussa, 4-naapurustoa käyttävässä ruudukossa polun pituus muodostuu Manhattan-etäisyyttä noudattaen. Polku on merkitty kuvaan punaisella, ja sininen viiva on lyhyin mahdollinen suora polku keltaisella merkittyjen lähdön ja maalin välillä. Polulla voi olla vain 90° kulmia, ja yksi lyhyimmistä reiteistä voidaan muodostaa tekemällä vain yksi 90° käännös maalia kohti.

Kuvan 14 keskellä kohdassa b. on esitetty kuusioruudukossa muodostuva reitti kohdan a. kaltaisesti. Reitin pituus kuusioruudukolla on lyhyempi, sillä kuusioruudukolla polku lähestyy maalia sekä y- että x-akseleilla liikuttaessa viistoon. Liikkumisen kustannus on sama kaikkiin suuntiin yksikkökustanteisessa kuusioruudukossa, kuten aiemmin mainittiin. Kuvan oikean reunan kohdassa c. 8-naapurustoisessa neliöruudukossa viistoon liikkuminen lyhentää edelleen polkua. Tämä johtuu siitä, että viistoon liikkuminen vie polkua vielä lähemmäs maalia y- ja x-akseleilla. Tärkeää on kuitenkin muistaa, että kyseessä on yksikkökustanteinen ruudukko. Ruudukossa liikkuminen viistottain ei siis vastaa yksikkökustannusta, jos ruudukko peittää

alleen esimerkiksi kartan joka vastaa kaksiulotteista, karteesista avaruutta.

Bjornssonin ja muiden (2003) analyysin mukaan neliöruudukoidulla 4-naapurustolla IDA*-algoritmin hakusyvyyden ollessa D joidenkin etäisyyden d erottamille solmuille, kuusioruudukossa hakusyvyys on vain $0,81D$. 8-naapuruston tapauksessa hakusyvyys on $D\backslash\sqrt{2} \approx 0,71$. Yap (2002a) osoittaa, että tex-ruudukoinnilla haarautumiskerroin vastaa kuusioruudukkoa. Tämä on ilmeistä kuusioruudukon ja tex-ruudukon topologioiden vastatessa toisiaan. Tex-ruudukot ovat Yapin mukaan teoreettisesti vain hieman kuusioruudukkoa hitaampia IDA*-algoritmin tapauksessa. Jos heuristiikan vaikutusta ei huomioida, kompleksisuudeksi muodostuu kuusioruudukolla $O(2.42^D)$ ja tex-ruudukolla $O(2.43^D)$. Lisäksi Yap toteaa, että tex-ruudukoilla löytyvä polku on aina lyhyempi kuin 4-naapurustoisella neliöruudukolla, siinä missä kuusioruudukolla vastaava polku voi joissakin tapauksissa pidempi.

IDA*-algoritmia käytettäessä ruudukoinnin valintaprosessi on tärkeä algoritmin tehokkaan toiminnan varmistamiseksi. Kuusioruudukoilla haut ovat eksponentiaalisesti nopeampia kuin 4- tai 8-naapurustoisella neliöruudukolla. Kuusioruudukko on optimaalinen hakunopeuden kannalta kaikkien säännöllisten ruudukointien joukossa. Käytännön toteutuksissa kuusioruudukon hyödyt voidaan saavuttaa minimaalisella tehohäviöllä käyttämällä tex-ruudukointia, joka on helpompi toteuttaa. (Yap, 2002a) Bjornsson ja muut (2003) huomauttavat kuitenkin, että IDA*-algoritmin toiminnan analysoinnista satunnaisilla kartoilla ei voida vetää suoria johtopäätöksiä algoritmin toimintaan realistisilla kartoilla. Heidän havaintojensa mukaan IDA*-algoritmi oli huomattavasti nopeampi satunnaisilla kartoilla, kun taas A*-algoritmi toimi paremmin videopelikartoilla.

5.5. Kulmariippumaton reitinhaku

Edellä kuvattujen ruudukointien ja reitinhakualgoritmien tapauksessa syntyvä reitti seuraa käytetyistä ruudukoinnista muodostuvaa rakennetta. Esimerkiksi 8-naapurustoisella neliöruudukolla tämä johtaa siihen, että polku sisältää vain 45° tai 90° kulmia. Tällainen liikkuminen ei useimmilla kartoilla ole tarpeen. Esimerkiksi maastossa kulkeva henkilö voi tietenkin liikkua mihin tahansa suuntaan mistä tahansa pisteestä, eikä polulle sijoittuvia käännöksiä ole tarpeen keinotekoisesti rajoittaa. Kulmariippumattomat reitinhakualgoritmit pyrkivät etsimään sellaisia polkuja, joissa polun suunta voi muuttua millä tahansa kulmalla.

A*-algoritmilla muodostettuja polkuja voidaan jälkikäsitellä siten, että lopputulos on kulmariippumaton ja lyhyempi kuin ruudukkoa noudattava polku. Botea,

Müller ja Schaeffer (2004) käyttävät esimerkiksi menetelmää, jossa maalisolmusta lähtien palataan polkua taaksepäin poistaen solmuja. Jos löytyneellä polulla on jossakin kohdassa solmut $a \rightarrow b \rightarrow c$, solmu b poistetaan polulta, mikäli solmujen a ja c välille voidaan piirtää suora viiva siten, ettei viiva leikkaa mitään estettä. Toisin sanoen solmu poistetaan, jos sen viereisten solmujen välillä on näköyhteys.

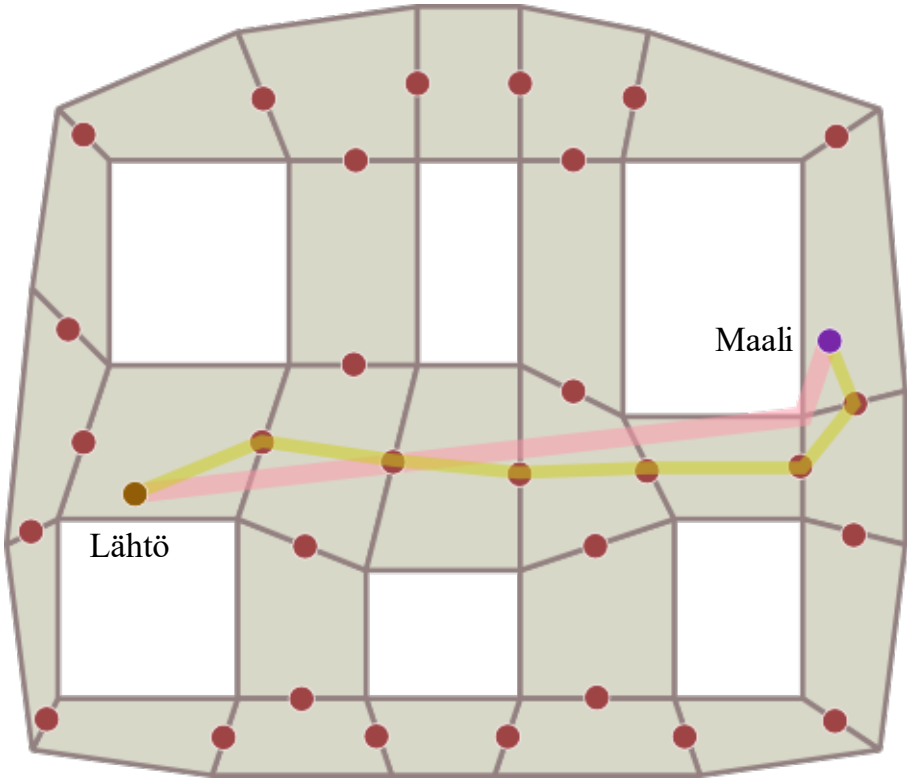
Nashin, Danielin ja Koenigin (2007) esittelemä Theta*-algoritmi on A*-algoritmin variantti, joka etenee ruudukon ruutujen reunoja pitkin muodostaen kulmariippumattomia polkuja. Keskeisenä erona on, että Theta*-algoritmi tarkastelee kunkin naapurisolmun tapauksessa myös suoraa näköyhteyttä edelliseen solmuun polulla. Jos solmujen välillä on esteetön yhteys, vanhemmaksi merkataan aina tämä edellinen solmu. Vastaavasti solmujen kustannukset arviointifunktion $\hat{g}(n)$ osassa lasketaan tämän suoran näköyhteyden etäisyyden perusteella. Tämän näköyhteyden tarkistaminen jokaisen solmun kohdalla on laskennallisesti raskas operaatio, mutta Nash et al. esittelevät myös menetelmän jolla laskenta voidaan suorittaa lineaarisesti ruutujen määrään nähden.

Nashin ja muiden (2007) havaintojen mukaan Theta*-algoritmi löytää huomattavasti lyhyempiä polkuja kuin A*. Vaikka haku toimii pääpiirteittäin samalla tavalla, löytyvästä polusta tulee lyhyempi Theta*-algoritmin kyetessä oikomaan polkua tilanteissa, joissa näköyhteys seuraavaan solmuun saavutetaan. Huomionarvoista kuitenkin on, että testatut ruudukot ovat yksikkökustanteisia. Vaihtelevat kustannukset tuovat kulmariippumattomaan reitinhakuun lisähaasteen, sillä kahden solmun välille näköyhteyttä laskiessa tulisi huomioida jotenkin suoran viivan alle jäävän alueen vaihteleva kustannus. Tämä ongelma ilmenee myös Botean ja muiden (2004) käyttämässä polun jälkiprosessoinnissa: vain yksikkökustanteisessa polussa voidaan kolmioepäyhtälön perusteella poistaa jokin solmu sen vierekkäisten solmujen välisen näköyhteyden perusteella.

Kulmariippumattomaan reitinhakuun soveltuvia algoritmeja ovat myös muun muassa Mendoncan ja Goodwinin (2015) C-Theta*-algoritmi sekä Haraborin ja Grastienin (2013) Anya-algoritmi. C-Theta*-algoritmi hyödyntää klusterointia vähentääkseen solmujen välisiin suoriin näköyhteyksiin tarvittavaa laskentaa. Sen tuottamat polut ovat lyhyempiä kuin A*-algoritmin, ja suorituskyvyltään C-Theta* on nopeampi kuin Theta*. Anya-algoritmin toiminta taasen perustuu kulloinkin tarkasteltavan hakuavaruuden jakamiseen intervalleihin ja näiden intervallien tarkasteluun kerralla. Harabor ja Grastien (2013) osoittavat Anya-algoritmin toimivan optimaalisesti, eikä se vaadi mitään ylimääräisiä tietorakenteita tai esiprosessointia toimiakseen.

5.6. Navigointiverkot

Edellä käsitellyt ruudukoinnit ovat muodoltaan säännöllisiä, eli kaikki ruudut ovat geometrialtaan samanlaisia. Kartta voidaan kuitenkin jakaa myös epäsäännöllisiksi muodoiksi, joiden muodostamassa topologiassa reitinhakualgoritmit voivat toimia. Tällaiset monikulmioista koostuvat navigointiverkot (*navigation mesh*) ovat yleisiä esimerkiksi videopeleissä. Navigointiverkon etuna on, että se saattaa joissakin tapauksissa vähentää huomattavasti tarvittavaa laskentaa, sillä navigointiverkko kykenee kattamaan laajemman alueen pienemmällä määrällä solmuja. Ruudukoinnin tapauksessa nämä solmut ovat siis ruutuja, ja navigointiverkon tapauksessa ne ovat monikulmioita.



Kuva 15. Lyhyin polku navigointiverkolla. (Patel, 2017c)

Kuvassa 15 on esitetty yksinkertainen navigointiverkko. Navigointiverkkoon on merkitty lähtö- ja maalipiste. Tummempi alue kuvaa kuljettavaa aluetta ja valkoinen navigointiverkon ulkopuolelle jäävää, kulkukelvotonta aluetta. Tällainen monikulmioiden muodostama verkko on mallinnettavissa graafiksi luvussa 5 käsittelemäni ruudukon kaltaisesti. Kunkin monikulmion osalta tulee siis tietää kyseisen monikulmion naapurit. Koska monikulmiot eivät ole säännöllisiä, eroavaisuudet niiden geometriassa tulee huomioida esimerkiksi etäisyyksiä laskettaessa. Näiden navigaa-

tioverkon monikulmioiden välillä voi olla vaihtelua esimerkiksi etenemiskustannuksissa. Yksittäisen monikulmion sisällä jokin etenemistä kuvaava attribuutti on sama, mutta monikulmioiden välillä se voi vaihdella. (Patel, 2017c)

Kuvassa 15 navigointiverkon monikulmioihin on merkitty niiden välisten sivujen keskipisteet, jota keltaisella merkitty reitti seuraa. Näitä voidaan kutsua navigointipisteiksi. Pinkillä värillä korostettu todellinen optimaalinen reitti. Reitti voidaan muodostaa käyttämällä esimerkiksi monikulmioiden keskipisteitä, sivujen keskipisteitä tai monikulmion kärkiä navigointipisteinä. Näitä voidaan myös yhdistellä siten, että reitinhaun polut voivat kiinnittyä vaikkapa sivujen keskipisteisiin tai monikulmioiden kärkiin reitin lyhentämiseksi. (Patel, 2017c)

Esimerkiksi Cui ja Shi (2012) esittelevät A*-algoritmia käyttävän menetelmän, jolla voidaan hakea reittejä navigointiverkoissa. A*-algoritmi kohdistetaan monikulmioiden joukkoon, ja tuloksena saadaan järjestetty joukko monikulmioita. Tässä joukossa vierekkäiset monikulmiot ovat kartalla topologisesti vierekkäin. Koska joukko alueita ei vielä ole polku, täytyy polku muodostaa tämän monikulmiojoukon perusteella. Tässä voidaan Cuin ja Shin tapaan käyttää esimerkiksi kolmiointia. Kolmioinnissa monikulmiot korvataan kolmioilla, ja reitti muodostetaan siten, ettei se kulje yhden kolmion yli kuin kerran. Tässä hyödynnetään erityistä Monosen (2010) esittelemää suppiloalgoritmia.

Van Toll ja muut (2016) vertailevat artikkelissaan lukuisia erilaisia menetelmiä navigointiverkon muodostamiseksi. Siinä missä ruudukoinnin muodostaminen on helppoa ruudukoinnin geometrian ollessa hyvin määritelty, navigointiverkkojen muodostaminen on huomattavasti hankalampaa. Ruudukkoa muodostettaessa ruutujen alle jäävää karttaa ei tarvitse mitenkään huomioida. Navigointiverkon tapauksessa monikulmioiden kokoa ja muotoa rajoittavat kartalla sijaitsevat paikkatietokohteet.

Navigointiverkko muodostuu esimerkiksi kuvan 15 tapauksessa suhteellisen yksinkertaiseksi. Kartalla on ollut vain kuusi suorakulmion muotoista estettä. Monimutkaisilla kartoilla navigointiverkon monimutkaisuus luonnollisesti kasvaa. Kuten esimerkiksi Van Tollin ja muiden (2016) suorittamasta vertailusta käy ilmi, eri algoritmien tarkkuus vaihtelee. Kaikki algoritmit eivät esimerkiksi välttämättä kata täysin aluetta, jolle navigointiverkkoa muodostetaan. Navigointiverkkojen tapauksessakin voidaan siis tasapainotella suorituskyvyn ja muodostuvan verkon tarkkuuden välillä.

Eräs mielenkiintoinen menetelmä navigointiverkkojen muodostamiseksi monimutkaisilla kartoilla on Halen (2011) esittelemä planaarinen adaptiivinen tilantäyt-

töalgoritmi (*planar adaptive space filling volumes, PASFV*). Algoritmi toimii laajentamalla tietyistä pisteistä alkavia alueita, kunnes ne täyttävät niille käytettävissä olevan tilan. Hale käyttää menetelmän kuvauksessa analogiaa mikroaaltouuniin asetettavista vaahtokarkeista, jotka ”laajenevat täyttämään saatavilla olevan vapaan konfiguraatioavaruuden.” Tällaisessa laajenemisprosessissahan kappaleen laajeneminen pysähtyy niissä pisteissä, joissa on jokin este. PASFV-algoritmi kasvattaa vastaavasti siemenpisteistä laajenevia alueita. Laajeneminen pysäytetään ennen kuin yksikään monikulmioista muuttuu konkaaviksi. Tämän jälkeen algoritmi asettaa tyhjille alueille uusia siemenpisteitä, joista kasvu jatkuu täyttämään alueen. Algoritmin toiminta pysähtyy, kun uusia siemenpisteitä ei voida enää asettaa ja alueita ei voida enää kasvattaa.

Kuten Hale (2011) osoittaa, hänen PASFV-algoritminsa tuottaa äärimmäisen hyviä navigointiverkkoja esimerkiksi A*-algoritmin reitinhakua varten. Polkujen pituudet ovat keskimäärin lyhyempiä ja käännöksiä tulee keskimäärin polulle vähemmän kuin esimerkiksi käyttämällä Hertelin ja Mehlhornin (1985) HM-algoritmia. Halen testitapauksissa polku on muodostettu käyttäen monikulmioiden keskipisteitä ja monikulmioiden sivujen keskipisteitä. Navigointiverkossa muodostettavaa polkua voidaan vielä jalostaa tästä eteenpäin kolmioinnin avulla. (Cui & Shi, 2012)

Edellä esitellyillä navigointiverkkomenetelmillä on selvästi vahvuutensa suhteellisen yksinkertaisilla kartoilla, joissa kuljettavuuden vaihtelevuus on pientä. Maastokartoilla kohteita on paljon ja niiden koko sekä muoto vaihtelevat huomattavasti. Kohteet ovat myös osittain päällekkäin. Kuten esimerkiksi Van Tollin ja muiden (2016) analyysista käy ilmi, kartan monimutkaisuus lisää syntyvän navigointiverkon monimutkaisuutta. Mikäli maastokartoilla haluttaisiin käyttää navigointiverkkoja, tulisi joitakin alueita kyetä samaistamaan edettävyyden osalta. Tämä mahdollistaisi navigointiverkon monikulmioiden laajenemisen riittävän isoiksi. Tällöin reitinhakualgoritmin hakuavaruus pienenee merkittävästi tavanomaiseen ruudukointiin nähden.

6 Aiempi tutkimus

Reitinhakua maastokartoilla on tutkittu suhteellisen vähän verrattuna esimerkiksi reitinhakuun tieverkostoilla. Reitinhakua vektoreista muodostetuilla rastereilla ja muilla ruudukoilla on tutkittu paljon, mutta maastokarttoja on käytetty aineistona vähän. Esimerkiksi videopeleissä tekoälyn ohjaamien hahmojen liikuttamiseksi tarvitsee löytää reitti hahmolle jollakin kartalla. Pääsääntöisesti nämä kartat ovat kuitenkin hyvin erilaisia reaalimaailman maastokarttoihin verrattuna. Maastokartoilla on huomattavasti enemmän kohteita eikä karttaa voida erityisesti suunnitella hahmojen liikettä silmällä pitäen.

Mitchell ja Keirsey (1984) ovat tutkineet, kuinka digitaalisilla maastokartoilla voidaan suunnitella reittejä autonomisesti liikkuvalle kulkuneuvolle. Heidän menetelmänsä (*BITPATH*) perustuu kartan ruudukointiin. Mitchell ja Keirsey esittelevät myös ruudukoinnin tuoman poikkeaman korjaukseen soveltuvan menetelmän. He huomauttavat, kuinka pelkästään A*-algoritmin käyttö Dijkstran algoritmin sijaan vähentää ruudukoinnin aiheuttamaa poikkeamaa. Mitchellin ja Keirseyn poikkeamankorjausmenetelmä hyödyntää Vossepoelin ja Smeuldersin (1982) etäisyysmetriikkaa, jonka avulla määritellään solmusta sen naapuriin liikkumisen kustannukselle kerroin. Tämä kerroin määräytyy toistaiseksi kyseiseen solmuun kuljetun polun rakenteesta. Mitchellin ja Keirseyn havaintojen mukaan tällä menetelmällä voidaan tehokkaasti vähentää ruudukoinnin lisäämää poikkeamaa todelliseen lyhyimpään polkuun nähden. He kuitenkin huomauttavat, että tämän menetelmän käyttöä tulee tutkia lisää optimaalisten parametrien selvittämiseksi.

Mitchell ja Keirsey (1984) käyttivät tutkimuksessaan aineistona Yhdysvaltain puolustusministeriön silloisen karttaviraston (*DMA, Defence Mapping Agency*, nykyään *NGA, National Geospatial-Intelligence Agency*) tuottamia maastokarttoja. Heidän algoritminsa toimii kaksiulotteisessa avaruudessa, mutta aineistoa voidaan laajentaa korkeusmallilla, jonka avulla voidaan välttää esimerkiksi jyrkänteitä tai muita alueita, joissa kulkuneuvo ei kykene etenemään turvallisesti. Ruudukoinnissa Mitchell ja Keirsey käyttävät neliöruudukointia 8-naapurustolla. Heidän mukaansa ruudukon etenemiskustannukset voidaan muodostaa käyttäen mitä tahansa käyttäjän määrittelemiä tietolähteitä. Artikkelissaan he ovat käyttäneet DMA:n tietokannasta johdettua kaksiulotteista 64-bittistä taulukkoa, jonka yksittäinen elementti kuvaa aluetta, jonka koko on $12{,}5\,\mathrm{m} \times 12{,}5\,\mathrm{m} = 156{,}25\,\mathrm{m}^2$. Kustakin tällaisesta ruudusta on tallennettu 64-bittiin korkeusarvo, maaston tyyppi, kuljettavuus ja muut mielenkiintoiset kohteet kuten padot, tiet ja sillat.

Puolustusministeriön karttaviraston maastokartta on sisältänyt siis jonkinlaisen arvion maaston kuljettavuudesta. Mekanisoitujen joukkojen liikkeen suunnitteluun tällainen kartta on ehdottomasti tarpeellinen, riippumatta siitä, käytetäänkö automaattista reitinhakua vai ei. Mitchellin ja Keirseyn (1984) käyttämällä aineistolla on siis huomattava etulyöntiasema esimerkiksi tavanomaisiin maastokarttoihin nähden. Maastokartat eivät yleensä sisällä mitään erikseen määriteltyä tietoa kuljettavuudesta, joten se on johdettava muista maastokartan kohteista. Merkittävää on myös, että Mitchellin ja Keirseyn menetelmä on kehitetty osaksi laajempaa autonomisen kulkuneuvon navigointijärjestelmää, johon kuuluu myös reaaliaikainen lähiympäristön kartoittaminen esimerkiksi tutkan avulla. Autonomisen kulkuneuvon liikettä suunniteltaessa ”näkyvällä alueella” liikkuminen on oma tutkimusalueensa, jossa tulee huomioida esimerkiksi kulkuneuvon mekaaniset rajoitteet ja tasapainon säilyminen erilaisissa maastokonfiguraatioissa.

Mitchell (1988) on myöhemmin tutkinut maastokartoilla reitinhaun ongelmaa laajemmin erityisesti vektorikarttojen näkökulmasta. Hän toteaa, että laskennallisen geometrian algoritmeja hyödyntämällä voidaan ratkaista reitinhaun ongelma maastokartoilla. Korkeammalla tasolla tarkasteltuna maastokartta on vain kokoelma geometrisia muotoja, joten siitä voidaan mallintaa geometriseen reitinhakuun soveltuva ongelma. Mitchell käsittelee erityisesti sitä, kuinka maastoa tulisi mallintaa tehokkaiden, optimaalisia reittejä tuottavien algoritmien käyttöön. Hän käsittelee tavanomaisen ruudukoinnin lisäksi esimerkiksi nelipuuruudukoinnin, joka yhdistää vierekkäisiä samankaltaisia alueita suuremmiksi kokonaisuuksiksi.

Mitchellin (1988) myöhempi tutkimus eroaa Mitchellin ja Keirseyn (1984) aiemmasta tutkimuksesta hyödyntämällä vektorikarttoja ruudukoinnin sijaan. Mielenkiintoisena yksityiskohtana Mitchell esittelee, kuinka optiikasta tuttua Snellin taittumislakia voidaan hyödyntää reittiä laskettaessa. Optimaalisen reitin siirtyessä painotetulta alueelta toiselle voidaan reitin etenemiskulman muutos laskea alueiden painotusten sekä reitin tulo-ja taitekulmien suhteen avulla. Mitchellin mukaan Snellin taittumislakia hyödyntämällä poluista saadaan lokaalisti optimaalisia tietylle painotettujen alueiden reunaviivojen sekvenssille. Tämän reunaviivojen sekvenssin valinta ei kuitenkaan ole yksinkertaista.

Mitchell (1988) mainitsee vektorikarttojen eduksi niiden pienen koon ja mahdollisuuden kuvata kohteita äärimmäisen tarkasti. Hän esittelee, kuinka vektorikartasta voidaan muodostaa reitinhakuun soveltuva graafi määrittelemällä kartan geometriset oliot solmuiksi ja johtamalla niiden topologiasta näiden solmujen välille kaaria. Tämä muistuttaa luvussa 5 esittelemääni ruudukon ja graafin välistä

suhdetta. Keskeinen ero on se, että vektorikartoilla on ruutujen sijaan monikulmioita ja mahdollisesti muitakin geometrisia muotoja pelkkien neliöiden tai esimerkiksi kuusioiden sijaan. Tällaisella vektorikartasta muodostetulla graafilla reitinhakualgoritmin löytämä lyhyin polku on sekvenssi alueita, joiden läpi polku kulkee. Kuten kohdassa 5.6 mainitsin, tällaista aluesekvenssiä tulee vielä jalostaa esimerkiksi kolmioinnin avulla optimaalisen polun löytämiseksi. Lisäksi Mitchellin (1988) mukaan tällä aluesekvenssillä ei välttämättä ole mitään yhteneväisyyttä todellisen optimaalisen polun kanssa. Hän kuitenkin toteaa, että erityisesti epätarkoilla aineistoilla ja subjektiivisilla alueiden painotuksilla tämä tekniikka voi olla hyvinkin hyödyllinen.

Mitchell ja Papadimitriou (1991) ovat myöhemmin keskittyneet vektorimuotoisiin karttoihin ja niihin sovellettaviin geometrisiin algoritmeihin. Painotettujen alueiden euklidiseen reitinhakuun Mitchell ja Papadimitriou ovat esitelleet polynomisen algoritmin. Se sallii mielivaltaisen kokonaislukupainotuksen alueille väliltä $[0, +\infty]$. Monikulmioista koostuva alue on algoritmia varten kolmioitu, eikä Mitchellin ja Papadimitrioun menetelmä ota kantaa kolmioinnin toteutustekniikkaan tai kompleksisuuteen. Myös Richbourg, Rowe, Zyda ja McGhee (1987) ovat tutkineet vektorikarttojen käyttöä reitinhaussa. Heidän menetelmänsä nojaa myöskin Snellin taittumislakiin, jonka avulla kartasta johdetaan graafi, johon voidaan kohdistaa reitinhaku A*-algoritmin avulla.

Douglas (1994) on tutkinut reitinhaun ongelmaa maastokartoilla käyttäen neliöruudukointia ja 8-naapurustoa. Hän esittelee eri keinoja korkeusmallin ruudukointiin ja mainitsee, kuinka ruudukoilla voidaan interpoloida korkeusarvoja mille tahansa välille. Douglasin mukaan ruudukoidun korkeusmallin käyttäminen sellaisenaan kahden pisteen välisen näkyvyyden arvioimiseen on hulluutta, ja interpolaatio on näissä tapauksissa aina tarpeen. Hänen menetelmässään näkyvyyttä ei tarkastella, mutta interpolaatiota tehdään silti kustannusten laskennassa. Douglas tallentaa korkeusarvot ruudukon viivojen risteyspisteisiin ja ruudukon sisällä sijaitsevan pisteen korkeus interpoloidaan ruudun neljän nurkkapisteen korkeuksien avulla.

Douglasin (1994) menetelmässä yksittäisten ruutujen etenemiskustannuksista muodostetaan kertymäkustannusruudukko, joka kuvaa maalipisteeseen liikkumisen kustannuksia jokaisesta ruudusta. Tämä kertymäkustannusruudukko muodostetaan maalipisteestä lähtevällä algoritmilla, joka laajenee joka suuntaan täyttäen lopulta koko ruudukoidun alueen. Algoritmi tutkii kunkin ruudun 8-naapurustoa ja etenee pienimmän kustannuksen suuntaan. Edetessään se määrittelee tuntemattomien ruutujen kustannukset halvimman naapurin perusteella. Douglasin mukaan empiirinen tutkimus osoittaa, että ortogonaalinen naapurusto kannattaa tutkia ennen diago-

naalista naapurustoa.

Douglas (1994) painottaa artikkelissaan sitä, kuinka reitinhaun tulisi toimia optimaalisesti yhden yksinkertaisen muuttujan kanssa ennen useampien muuttujien käyttöä. Hänen havaintojensa mukaan useat senaikaiset kaupalliset sovellukset tuottivat selkeästi virheellisiä polkuja käytettäessä yhtä yksinkertaista kustannusmuuttujaa. Polut ovat silmämääräisesti vakuuttavia käytettäessä useampia muuttujia, sillä optimaalisen polun arviointi useamman muuttujan tapauksessa on hankalampaa. Esimerkiksi maastokartoilla reitinhakua tulisi hänen mukaansa ensin tutkia vaikkapa pelkästään suo- tai järvialueiden perusteella muodostetulla kustannusruudukolla ennen muiden tasojen lisäämistä reitinhakuun.

Rees (2004) on tutkinut reitinhaun ongelmaa polkujen näkökulmasta. Hän pyrkii selvittämään, millä perusteella ihmisten muodostamien polkujen reitti määräytyy Walesin vuoristoisella alueella. Rees käyttää alueen korkeusmallia ja Dijkstran algoritmia reittien muodostamiseen. Hän esittelee yksinkertaisen spatiaalisen kustannusfunktion, joka mallintaa ihmisen liikkumiskykyä erilaisilla kaltevilla pinnoilla. Mielenkiintoisena yksityiskohtana hänen kustannusfunktionsa on neliöllinen korkeuseron suhteen, joten muodostuvat polut etenevät siksakkia jyrkissä ylä- ja alamäissä. Kuten Rees toteaa, maastoon ihmisten liikkumisen seurauksena muodostuvat polut noudattavat useimmiten tällaista siksakkikuviota.

Rees (2004) on testannut algoritmiaan koneellisesti generoidulla korkeusmallilla, joka kattaa 5 km × 5 km alueen. Neliön muotoisten ruutujen koko on tässä synteettisessä korkeusmallissa 50 m × 50 m. Korkeusmalli on muodoltaan yksinkertainen; sen keskellä on kohouma, jonka rinteet ovat yhtä jyrkkiä joka suuntaan. Lisäksi Rees on soveltanut algoritmiaan korkeusmalliin, johon on lisätty satunnaista kohinaa. Hän osoittaa, kuinka kustannusfunktion parametrit vaikuttavat siksakin muodostumiseen rinteessä. Reesin mukaan algoritmin löytämä polku mutkittelee rinteessä mahdollisimman pienellä amplitudilla, vaikkakin todellisilla poluilla tätä amplitudia ei ole rajoitettu.

Synteettisen korkeusmallin lisäksi Rees (2004) on soveltanut menetelmäänsä myös reaalimaailmasta johdettuun korkeusmalliin. Korkeusmalli kattaa Walesin Snowdon vuorta ympäröivän alueen 9 km × 12 km laajuudelta 50 m × 50 m ruudukkokoolla. Alueella on paljon olemassa olevia polkuja, joihin Rees pyrkii vertaamaan hänen menetelmänsä tuloksia. Kustannusfunktion avulla muodostetut polut noudattavat todellisia polkuja tietyllä parametrijoukolla, joten selvästikin maastoon ihmisten muodostamat polut ovat jossakin määrin optimaalisia. Rees toteaa, että hänen löydöksensä ovat linjassa vuoristoisella alueella liikkumisen metabolisten kustannusten

tutkimuksen kanssa. Hänen tutkimuksensa käytti pelkästään korkeusmallia, mutta kuten Rees mainitsee, menetelmään voidaan lisätä myös maaston muut kohteet kustannuskertoimina.

Balstrøm (2002) on myös tutkinut lyhyimpiä reittejä vuoristoisella alueella. Hänen tutkimuksensa tavoitteena oli selvittää, miten Färsaarilla sijaitsevien sademittarien luona voidaan vierailla käyttäen mahdollisimman lyhyitä polkuja. Balstrøm käyttää aineistonaan karttojen lisäksi mitattuja todellisia kävelyaikoja erilaisilla rinteillä. Sademittarien tietojen keräämisen ongelma on ilmennyt 1980-luvulla, joten tietojen keruu on jouduttu tekemään käymällä mittarien luona. Nykyiset tutkimuksessa käytettävät sademittarit on useimmiten varustettu etäluennalla. Balstrømin mukaan maastossa paljon liikkuvien opiskelijoiden määrittelemiä reittejä pitkin tietojen keruu kesti keskimäärin viisi tuntia ja reitit olivat raskaita kulkea.

Balstrømin (2002) menetelmä perustuu alueen ruudukointiin ja ruutujen etenemiskustannusten muodostamiseen aineiston perusteella. Ruutujen kustannukset määräytyvät korkeuserojen, jokien, järvien ja mitattujen etene miskustannusten perusteella. Balstrøm laskee jokaisesta 17 sademittarista lyhyimmän polun muihin mittareihin. Tästä voidaan edelleen laskea reitti, joka kiertää kaikki 17 sademittaria käyttämällä mahdollisimman lyhyttä polkua kunkin mittarin välillä. Balstrømin mukaan polkujen raskaus on enimmäkseen seurausta siitä, että polut kulkevat jyrkkiä rinteitä ja ylittävät matalia jokia. Balstrøm mainitsee, että maaston tyypin huomiointi etenemiskustannuksissa osoittautui hankalaksi, sillä pienistä alueista ei voida ekstrapoloida koko alueen kattavaa informaatiota.

Ruudukon ruutujen kustannukset ovat Balstrømin (2002) tutkimuksessa reaalimaailman yksikössä. Yksittäisen ruudun kustannus on ruudun läpi kulkemiseen kuluva aika sekunteina. Yleisesti ottaen kustannuksena voidaan käyttää mitä tahansa yksikköä, kunhan yksikkö on sama kaikkien kustannusten osalta. Balstrøm mainitsee, että kustannuksena voitaisiin käyttää myös hapenkulutusta. Ihmisen hapenkulutusta on mitattu erilaisilla kantamuksilla ja ylä- sekä alamäissä, joten kustannusmallin muodostaminen on ainakin teoriassa mahdollista.

Balstrømin (2002) korkeusmallin ja vesistöjä kuvaavan rasterin ruudukkokoko on $5\,\mathrm{m} \times 5\,\mathrm{m}$. Koska hän on määritellyt sekä korkeusmallin että vesistöjen etenemiskustannukset sekunteina, voidaan nämä ruudukot yksinkertaisesti summata kokonaiskustannuksen muodostamiseksi. Itse reitti on muodostettu tässä summaruudukossa käyttäen ArcView-ohjelmiston Costpath-algoritmia. Kaikkien sademittarien välille muodostettujen lyhyimpien polkujen joukosta Balstrøm on valinnut kaikki sademittarit kiertävän polun käyttäen jotakin verkkoanalyysisovellusta. Ruudukoinnin

avulla muodostetut polut muodostavat yksinkertaisen graafin, jossa sademittarit ovat solmuja ja polut näiden välillä muodostavat graafin kaaret. Lisäksi polkujen risteyspisteet tulee tulkita solmuiksi, sillä näissä kohdissa voidaan vaihtaa polulta toiselle. Kaarien paino muodostuu polun pituudesta. Tällaiseen graafiin voidaan soveltaa mitä tahansa optimaalisten polkujen etsintään tarkoitettua algoritmia, kuten esimerkiksi luvussa 5 esiteltyjä Dijkstran algoritmia ja A*-algoritmia.

Balstrøm (2002) toteaa lopuksi, että ihmisen liikkuessa vaihtelevassa maastossa reitinvalintaan liittyviä tekijöitä on vaikea määrittää. Arvio maaston kuljettavuudesta on hyvin subjektiivinen ja ihmiset ovat fysiikaltaan erilaisia. Balstrømin mukaan tavanomaisessa patikoinnissa voi usein olla hyödyllistä kiivetä aluksi korkealle, jotta voidaan luoda silmämääräinen arvio ympäröivän maaston kuljettavuudesta ja lyhyimmästä reitistä. Tämän jälkeen reitti tyypillisesti etenee harjanteita ja valuma-alueita pitkin. Balstrøm mainitsee, että reitinhakualgoritmia voisi mahdollisesti muokata toimimaan tällä tapaa. Muutenkin hänen mukaansa ihmisten ja eläinten muodostamien polkujen ja reitinvalintaperusteiden tutkiminen voisi olla hyödyllistä automaattisen reitinhaun tehostamiseksi.

Reitinhakua on tutkittu aiemmin myös Suomen maastossa. Antikainen (2013) on hierarkkista A*-algoritmia (*HPA*, hierarchical pathfinding A**) hyödyntämällä pyrkinyt hakemaan tehokkaasti reittejä rastereilla, joissa kustannukset vaihtelevat. Osana tutkimustaan hän on toteuttanut menetelmän lisäosana ArcGIS-paikkatietojärjestelmään ja testannut sitä Keski-Suomesta otetuilla satelliittikuvilla. Satelliittikuvista on muodostettu kuljettavuusrasteri luokittelun avulla. Testausta varten Antikainen on valinnut kolme satunnaista aluetta, joissa ruudukkokoko on $30\,\mathrm{m} \times 30\,\mathrm{m}$. Alueet koostuvat 500×500 ruudusta, eli kukin niistä kattaa $255\,\mathrm{km}^2$ alueen.

Hierarkkisella A*-algoritmilla Antikainen (2013) pyrkii pienentämään reitinhaun hakuavaruutta. HPA*:n tehokkuus perustuu abstrahointiin. Kartta jaetaan samansuuruisiksi lohkoiksi ruudukoinnin tapaisesti. Näiden lohkojen reunaviivoista etsitään siirtymäkohdat vierekkäisten lohkojen välille. Tämä lohkorakenne on siis tarkkaresoluutioisen rasterin päällä korkeampana abstraktiotasona. Antikaisen mukaan siirtymillä on suuri merkitys algoritmin tuottamien reittien laadun suhteen. Hän keskittyykin käsittelemään ja vertailemaan kolmea eri strategiaa siirtymien sijoittamiseen lohkoihin.

HPA*-algoritmilla reitti minkä tahansa kahden pisteen u ja v välille haetaan tutkimalla ensin lohkoja, jotka sisältävät pisteet u ja v. Jos pisteet sijaitsevat samassa lohkossa, reitinhaku on triviaalia lohkojen suhteellisen pienestä koosta johtuen. Jos pisteet sijaitsevat eri lohkoissa, etsitään pisteistä reitti kyseisten lohkojen

reunoilla sijaitseviin siirtymäkohtiin. Tämän jälkeen lopullinen reitti voidaan hakea lohkotasolla käyttämällä A*-algoritmia. Koska lohkot ovat suuria tarkkaresoluutioiseen rasteriin nähden, A*-algoritmin hakuavaruus on pieni. Lohkoista muodostuvan reitin löydyttyä tarvitsee vielä hakea reitti kunkin lohkon sisällä. Koska lohkotasolla reitti kulkee lohkojen siirtymäkohtien kautta, lohkon sisällä reitti haetaan siirtymäkohdasta toiseen. HPA*-algoritmi siis poissulkee tarkkaresoluutioisesta rasterista suuren määrän ruutuja hyödyntämällä muodostettua korkeampaa abstraktiotasoa.

Lohkojen koolla on suuri vaikutus algoritmin laskennalliseen kompleksisuuteen. Lohkokoon kasvaessa reitti lohkon sisällä sijaitsevasta mielivaltaisesta pisteestä lohkon siirtymäkohtiin pitenee. Antikaisen (2013) mukaan lohkokoon vaikutusta voidaan pienentää käyttämällä useampia eri lohkokokoja, ja muodostamalla näistä monitasoisia graafeja. Tällöin pienempiä lohkokokoja voidaan hyödyntää suuremman lohkon sisällä etsittäessä reittiä pisteestä siirtymäkohtaan, eikä hakuja ole tarpeen tehdä tarkkaresoluutioisessa rasterissa.

Antikainen (2013) toteaa, että HPA*-algoritmin käyttämisessä on omat ongelmansa, vaikka se nopeuttaakin reitinhakua. Abstrahoinnista johtuen löytyvät reitit poikkeavat joiltakin osin optimaalisesta reitistä. Antikaisen mukaan tuloksia voidaan parantaa lisäämällä siirtymiä lohkojen välille, mutta tällöin laskennalliset kustannukset kasvavat. Lopuksi hän toteaa HPA*-algoritmia käyttävän menetelmän soveltuvan hyvin reitinhakuun maastossa, jos käytössä on vähän laskentaresursseja ja -aikaa. Abstrahoinnin vaatiman suhteellisen raskaan esiprosessoinnin takia menetelmä sopii erityisesti tilanteisiin, joissa lasketaan jatkuvasti reittejä saman alueen sisällä.

Kuten edeltä huomataan, reitinhaun ongelmaa tieverkostojen ulkopuolella on tutkittu sekä rasteri- että vektorikarttojen näkökulmasta. Näitä kahta lähestymistapaa ovat vertailleet Van Bemmelen ja muut (1993) artikkelissaan, joka esittelee myös laajennetun rasterin (*extended raster).* Laajennettu rasteri perustuu graafin muodostamiseen ruudukosta siten, että solmujen sijainnin katsotaan olevan ruudun reunalla. Lisäksi solmuja voi olla ruudun reunaviivalla useita, jolloin reitinhaku voi edetä yksittäisen ruudun läpi monessa eri kulmassa. Van Bemmelen ja muut esittelevät myös, kuinka naapurusto voidaan laajentaa kattamaan jopa 128 naapuriruutua myös tavallisilla ruudukoilla.

Van Bemmelen ja muut (1993) käyttävät vertailussaan rasterikarttojen tapauksessa Dijkstran algoritmia. He tutkivat, kuinka erilaiset naapurustot ja laajennettu rasteri vaikuttavat polun vääristymiseen. Neliöruudukoinnin lisäksi Van Bemmelen ja muut käyttävät myös nelipuuta rasterin muodostamiseksi. Vektorikarttojen rei-

tinhakuun he ovat toteuttaneet algoritmin, joka muodostaa rajoittuneen Delaunayn kolmioinnin (*constrained Delaunay triangulation, CDT*) vektorikartan kattamasta alueesta. Kolmioinnin avulla voidaan hyödyntää Snellin taittumislakia laajentamalla lähtöpisteestä hakurintamaa, joka taittuu alueen reunan kohdatessaan alueiden etenemiskustannusten ja tulokulman perusteella.

Van Bemmelen ja muut (1993) ovat toteuttaneet rasteri- ja vektorialgoritmit C++-kielellä, käyttäen erillisiä kirjastoja geometrisiin tietorakenteisiin ja abstrakteihin datatyyppeihin. Heidän toteutuksensa tavoittelee hyvää absoluuttista suorituskykyä, siinä missä esimerkiksi Antikaisen (2013) tavoitteena oli kyetä vertailemaan eri strategioita siirtymäkohtien sijoitteluun. Antikainen mainitsee, että 225 km^2 kokoisen alueen esiprosessointi kestää joillakin strategioilla useita tunteja. Reitinhaku alkuperäisellä rasterilla ilman abstrahointia vei Antikaisen mukaan jopa puoli tuntia hänen toteuttamallaan menetelmällä, siinä missä ArcGIS:n sisäänrakennetut reitinhakualgoritmit suoriutuivat samasta tehtävästä sekunneissa. Käytettävä laitteisto vaikuttaa myös merkittävästi eri toteutusten suoritusnopeuteen, joten niiden keskinäinen vertailu ei ole kovinkaan mielekästä.

Van Bemmelen ja muut (1993) toteavat, että rastereihin perustuvien menetelmien selkeä etu on niiden yksinkertaisuus. Rasterimenetelmät ovat helppoja toteuttaa, ja ne suoriutuvat reitinhausta kohtuullisen nopeasti. Rasterista löydetyt reitit kuitenkin poikkeavat todellisesta lyhyimmästä reitistä ruudukoinnin aiheuttaman vääristymän takia. Reittien laatu paranee rasterin resoluution ja reitin kääntymiskulmien määrän lähestyessä ääretöntä, mutta käytännössä laskennalliset rajat tulevat vastaan suhteellisen nopeasti. Vektorimenetelmillä saadaan tarkkoja tuloksia, siinä missä rasterilla löydetty reitti on vain likimääräinen arvio todellisesta lyhyimmästä reitistä. Tarkkojen tuloksien saavuttamisen hintana on kuitenkin merkittävästi suuremmat laskennalliset kustannukset. Van Bemmelen ja muut toteavatkin, että he eivät ole tyytyväisiä toteutetun vektorimenetelmän tehokkuuteen ja ehdottavat lopuksi muutamia potentiaalisia keinoja laskennan tehostamiseksi.

Selvästikin reitinhaku maastokartoilla voidaan toteuttaa kahdella eri lähestymistavalla, käyttäen joko vektoreita tai rastereita. Rasterilähestyminen on vaihtoehdoista epätarkempi, mutta se on suhteellisen yksinkertainen ja tehokas toteuttaa. Erilaiset hybridiratkaisut kuten esimerkiksi Van Bemmelenin ja muiden (1993) mainitsemat diskreetit vektorit pyrkivät yhdistelemään rasterien ja vektorien parhaita puolia tasapainoisen menetelmän saavuttamiseksi. Paikkatietojärjestelmien käyttäjämäärien lisääntyessä jatkuvasti reitinhakumenetelmiin kohdistuu edelleen paljon tutkimusta, ja laskennallisen geometrian saavutukset esimerkiksi painotettujen dis-

kreettien alueiden reitinhaussa tuovat paikkatietojärjestelmien käyttöön yhä tehokkaampia algoritmeja.

7 Reitinhaku maastotietokannasta

Osana tätä tutkielmaa toteutin sovelluksen, joka hyödyntää edellä käsittelemiäni menetelmiä reitinhakuun Maanmittauslaitoksen maastotietokannasta. (Karjalainen, 2017) Valitsin toteutettavaksi menetelmäksi A*-algoritmin rasterikartalla. Rasterimenetelmä on yksinkertainen ja sen muisti- sekä suoritusaikavaatimukset ovat kohtuulliset. Maastossa liikuttaessa toteutuva reitti on harvoin suora. Maastossa on paljon karttaan merkitsemättömiä kohteita, kuten kaatuneita puita ja ampiaispesiä. Suuremmalla ruudukkokoolla rasterimenetelmän suoritusaika lyhenee merkittävästi, kuten aiemmin on todettu. Tällöin rasterimenetelmää voidaan käyttää tehokkaasti korkeamman tason reittien hakemiseksi. Lopullinen reittivalinta jää käyttäjän vastuulle. Sovelluksen käyttäjä voisi tässä tapauksessa olla ihminen tai vaikkapa autonominen ajoneuvo.

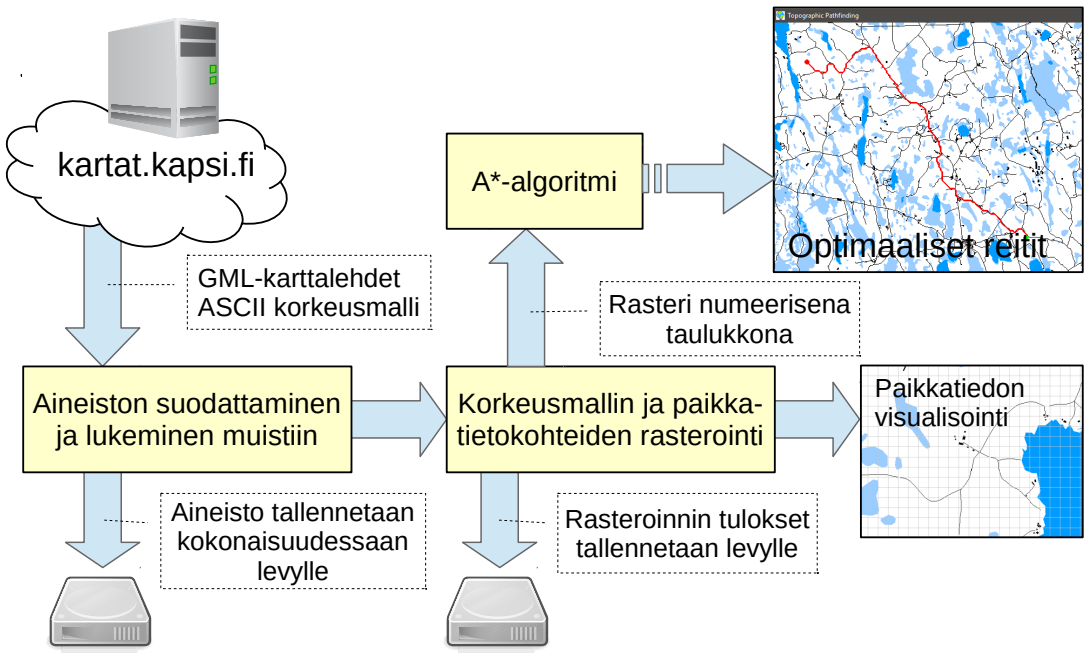
Käytän tässä luvussa esimerkkinä reittien suunnittelua jalkaisin maastossa kulkevalle. Sovellus voidaan konfiguroida käyttämään useita eri maastotietokannan kohderyhmiä, joita käsittelen tarkemmin kohdassa 7.2.1. Lisäksi kutakin kohderyhmää varten voidaan määritellä menetelmä kohteiden rasteroimiseksi. Sovellus kykenee siis laskemaan reittejä moniin eri tarkoituksiin, vaikka tässä luvussa käsittelenkin vain osaa kohderyhmistä lyhyesti. Lisäksi valitsemani kohderyhmät eivät varmastikaan ole paras mahdollinen maaston kuljettavuutta kuvaava osajoukko. Maaston kuljettavuuden määrittelyn olen jättänyt tämän tutkielman ulkopuolelle. Kuljettavuuden mittaaminen on toki mahdollista, mutta vaatii mittavia järjestelyjä. Maastotietokohteista voidaan kuitenkin valita ainakin selkeät esteet, kuten järvet ja suot. Samoin karttaan voidaan lisätä tieviiva-kohteet, jolloin reitinhaku ulottuu myös polkuihin.

Ainoa sovellukselle asettamani tavoite oli kyky hakea koordinaattipisteistä muodostuva optimaalinen reitti kahden käyttäjän määrittelemän pisteen välille. Myöhemmin laajensin tavoitteita kattamaan myös konfiguroitavuuden. On ilmeistä, että reitinhakuun ei ole mitään yksikäsitteisiä, optimaalisia parametreja. Maaston kuljettavuus on subjektiivista, mutta esimerkiksi suunnistajien etenemisnopeuksien perusteella voitaisiin muodostaa nykyistä objektiivisempi malli. Sovelluksen laskemien reittien optimaalisuus on siis suhteellista. Niiden voidaan todeta olevan käytettyjen parametrien suhteen optimaalisia, sillä reitinhakualgoritmin käyttämä heuristiikka on hyväksyttävä.

Tässä luvussa sovellukseen liittyvät laskenta-ajat on mitattu 4,08 gigahertsin taajuudelle kellotetulla Intelin Core i5-3570K -suorittimella. Käytetyssä tietokoneessa on keskusmuistia 16 gigatavua, joten resursseja sekä muistin että laskentate-

hon suhteen on kiitettävästi. Käyttöjärjestelmänä sovelluksen ajoissa on ollut 64-bittinen Windows 10 Pro. Mittaustilanteessa on käytetty sovelluksen oletusarvoisia rasterointi- ja kustannuskonfiguraatioita, `default_rasterize_parameters.json` ja `default_cost_parameters.json`.

Laskenta-ajoista voidaan muodostaa suuntaa antava arvio siitä, kuinka tehokkaasti reitinlaskentaa voitaisiin todellisissa sovelluksissa tehdä. Koska paikkatiedon esiprosessointivaihe vaatii paljon laskentaa, tällaisen sovelluksen potentiaali lienee laskennan ulkoistamisessa erilliselle palvelimelle. Esiprosessoinnin tulokset voidaan tallentaa välimuistiin, jolloin peräkkäiset reitinhaut samalle alueelle voidaan käsitellä nopeasti. Lisäksi esimerkiksi maastossa liikkuvalla on usein käytössään vain jonkinlainen mobiililaite, jonka laskentateho on rajattu. Erilliselle palvelimelle ulkoistettu reitinhaku takaa tehokkuuden päätelaitteesta riippumatta, joskin tällöin käyttäjä on riippuvainen internet-yhteydestä.



Kuva 16. Toteuttamani sovelluksen toiminnallinen rakenne.

Kuvassa 16 on esitetty toteuttamani sovelluksen toiminnallinen rakenne. Sovellus lataa maastotietokannan aineistoa Kapsi ry:n palvelimelta. Aineisto tallennetaan levylle ja konfiguraation perusteella osa paikkatietokohteista ladataan muistiin. Tämän jälkeen sovellus rasteroi valitut paikkatietokohteet ja korkeusmallin. Rasteroinnin tulokset tallennetaan levylle, jotta saman aineiston esiprosessoinnilta vältytään jatkossa. Sovellus piirtää paikkatiedon perusteella kartan, josta käyttäjä voi määri-

tellä lähtö- ja maalipisteen reitinhakua varten. A*-algoritmia sovelletaan puhtaasti numeeriseen taulukkoon. Korkeusmallin osalta algoritmi huomioi liikkumissuunnan rasterilla. A*-algoritmi tuottaa parametrien puitteissa optimaalisia reittejä, jotka sovellus edelleen piirtää kartalle. Sovelluksen yksityiskohtia käsittelen tarkemmin seuraavissa kohdissa.

7.1. Käytetyt työkalut

Kuten kohdassa 2.3 mainitsin, useimmat paikkatietosovellukset tarjoavat mahdollisuuden niiden laajentamiseen eri ohjelmointikielillä. Reitinhaun toteuttavan sovelluksen voi siis toteuttaa joko osaksi olemassa olevaa paikkatietosovellusta tai kokonaan omana sovelluksenaan. Osana valmista paikkatietosovellusta reitinhaku hyötyy joistakin isäntäsovelluksen ominaisuuksista. Esimerkiksi valmius käsitellä useita eri karttaformaatteja ja suorittaa muunnoksia näiden välillä mahdollistaa useampien aineistojen käyttämisen. Toisaalta osana valmista sovellusta isäntäsovelluksen arkkitehtuuri ja tekniset yksityiskohdat voivat rajoittaa laajennoksien toimintaa.

Päädyin toteuttamaan reitinhakusovelluksen erillisenä sovelluksena, sillä totesin työmäärän olevan samaa luokkaa molemmilla edellä mainituilla lähestymistavoilla. Sovelluksen toteuttaminen laajennoksena johonkin paikkatietosovellukseen edellyttäisi perehtymistä sovelluksen dokumentaatioon ja lähdekoodiin. Erillisen sovelluksen tapauksessa aikaa vie vastaavasti valittuihin kolmannen osapuolen kirjastoihin tutustuminen ja suunnittelutyö. Koin, että erillisen sovelluksen toteuttaminen johtaa perusteellisempaan tutustumiseen paikkatiedon käsittelyyn. Mahdollisimman suora pääsy aineistoon ja mahdollisuus vaikuttaa aineiston käsittelytapaan tarjoavat myös mahdollisuuksia sovelluksen toiminnan optimoimiseen. Paikkatietoindeksien ja -rakenteiden käyttö jää erillisessä sovelluksessa yksinomaan suunnittelijan vastuulle. Järjestelmän kaikki osat voidaan siis suunnitella reitinhakua tukevaksi.

Toteutin sovelluksen Pythonin versiolla 2.7. Valitsin paikkatiedon käsittelyyn GDAL-kirjaston (*Geospatial Data Abstraction Library*) version 2.1.3. (Open Source Geospatial Foundation, 2017a) GDAL on todella monipuolinen kirjasto, joka kykenee mallintamaan useita eri tiedostoformaatteja yhden rasterirajapinnan ja yhden vektorirajapinnan läpi. GDAL sisältää myös lukuisia komentorivityökaluja, joilla voi käsitellä paikkatietoaineistoja ja suorittaa esimerkiksi muunnoksia eri formaattien välillä. Kirjaston dokumentaatio on pääosin laadukas. Se sisältää paljon esimerkkejä kirjaston käytöstä ja erilaisista käyttötapauksista. Python-rajapinnan dokumentaatio on osittain puutteellinen, mutta kirjaston toiminnan ymmärtämiseksi C++-

rajapintaan tutustuminen on välttämätöntä.

GDAL huomioi aineiston projektion ja käytetyn koordinaattijärjestelmän. Maastotietokannan tapauksessa koordinaattijärjestelmä on ETRS-TM35FIN. Tätä ei tarvitse erikseen huomioida, sillä GDAL osaa suorittaa esimerkiksi geometriset operaatiot käyttämällä oikeaa koordinaattijärjestelmää. Vaikka GDAL osaa käsitellä myös rasteritiedostoja, käytin sitä sovelluksessa pelkästään aineiston lukemiseen ja muutamien paikkatietokohteiden määrittelyyn. Korkeasta abstraktiotasostaan huolimatta GDAL vaikuttaa erinomaiselta kirjastolta paikkatiedon käsittelyyn sekä monimutkaisissa että yksinkertaisissa sovelluksissa.

Karttojen ja reittien piirtämiseen käytettäväksi kirjastoksi valitsin PyGamen version 1.9.3. (PyGame developers, 2017) Koska visualisointi on varsinaisen tavoitteen kannalta vähemmän oleellinen, pyrin valitsemaan kirjaston, jonka käyttö on suoraviivaista. PyGame on suhteellisen tehokas ja minimaalinen peliohjelmointiin keskittynyt kirjasto. Se sisältää toiminnallisuuden erilaisten kuvioiden piirtämiseen ja käyttäjän syötteen käsittelyyn. PyGamen selkein etu on sen yksinkertaisuus, vaikkakin esimerkiksi tuki sisäkkäisten monikulmioiden piirtämiselle puuttuu.

7.2. Maanmittauslaitoksen maastotietokanta

Maanmittauslaitoksen maastotietokanta kattaa koko Suomen. Se käsittää muun muassa liikenneväyläverkoston, rakennukset, vesistöt, hallintorajat ja maastonkohdat kuten kivet, kalliohalkeamat ja niityt. Aineistona maastotietokanta on siis valtava, ja sitä käytetään useiden erilaisten karttojen tuottamiseen. Maastotietokanta soveltuu myös erinomaisesti reitinhakusovelluksiin. Sitä käytetään lisäksi esimerkiksi rakentamisen suunnittelussa ja ympäristöön kohdistuvassa tutkimuksessa. (Maanmittauslaitos, 2017d) Tässä tutkielmassa käyttämäni maastotietokannan osat on lisensoitu Maanmittauslaitoksen avoimen tietoaineiston CC 4.0 -lisenssillä, ja ne on noudettu toukokuussa 2017. (Maanmittauslaitos, 2017c)

Maastotietokantaa päivitetään jatkuvasti erityisesti tiestön ja nimistön osalta. Hallintorajat ja rakennukset päivitetään tietokantaan vuosittain. Maaston osalta kohteita päivitetään karttalehdittäin 3-10 vuoden sykleissä. Maanmittauslaitos ylläpitää maastotietokantaa Smallworld GIS -paikkatieto-ohjelmistoon perustuvalla JAKO/MTJ-sovelluksella. Maastotietokanta on tarkin valtakunnallinen paikkatietoaineisto: sen sijaintitietojen tarkkuus vastaa mittakaavaa 1:5000 - 1:10 000. (Maanmittauslaitos, 2016) Maastotietokanta on puhtaasti vektoriaineistoa, mutta Maanmittauslaitoksen avoimiin aineistoihin kuuluu myös rasterimuotoisia kartta-

sarjoja. (Maanmittauslaitos, 2017d)

Maastotietokanta on ladattavissa Kapsi ry:n palvelimelta osoitteesta `http://kartat.kapsi.fi`. Maanmittauslaitoksen avoimien aineistojen tiedostopalvelun kautta on mahdollista ladata osia aineistoista, mutta automatisoitu lataus on hankalaa. Kapsi ry:n palvelimella Maanmittauslaitoksen aineistot on indeksoitu karttalehdittäin, ja niiden lataaminen on suoraviivaista. Kapsi synkronisoi kaikki aineistot kerran vuorokaudessa. (Kapsi Internet-käyttäjät ry, 2017)

Maastotietokanta on kokonaisuudessaan tarjolla kolmessa eri formaatissa: GML, MIF ja SHP. GML on kohdassa 2.3.1 esitellyn OSGeon avoin standardi. GML on XML-kielioppi, jonka tavoitteena on mahdollistaa paikkatietokohteiden helppo käsittely ja siirrettävyys. GML:n tarkan standardin lisäksi siihen on kehitetty skeemoja eri sovellusalueita varten. Näitä sovellusalueita ovat esimerkiksi lentosää ja kolmiulotteiset kaupunkimallit. (Open Geospatial Consortium, 2017a)

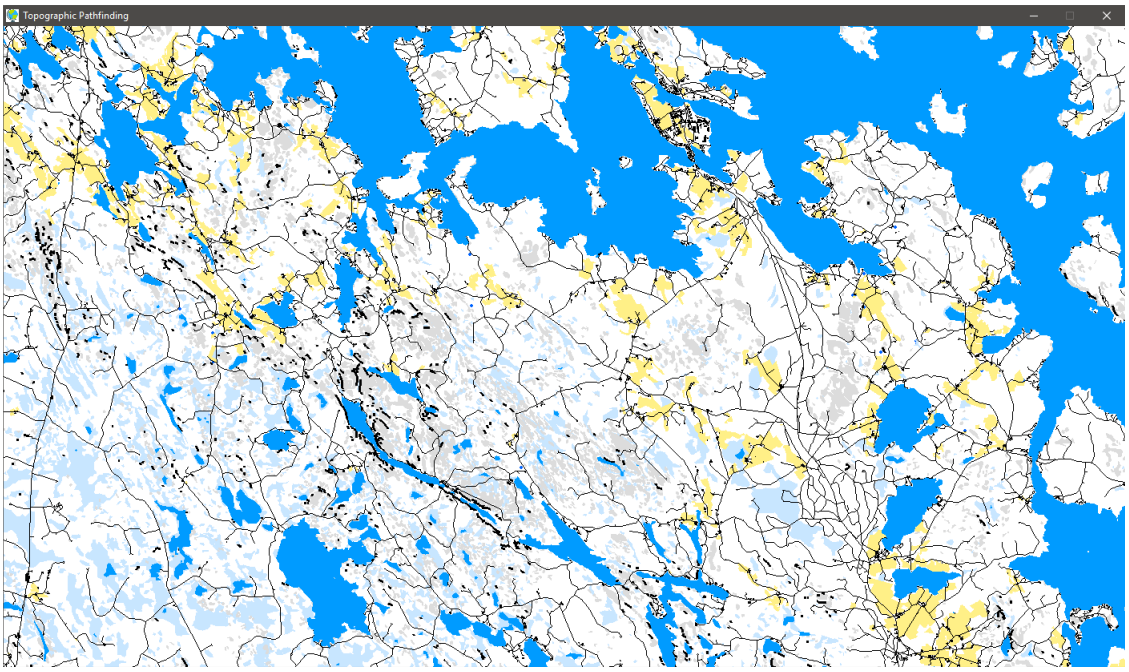
MIF on MapInfo-sovelluksen käyttämä suljettu vektoriformaatti. MapInfo käyttää sisäisesti TAB-formaattia. Siirrettäessä tietoja sovelluksen ulkopuolelle aineistot muunnetaan MIF-formaattiin. Karttalehdet koostuvat MIF-formaatissa erillisistä MIF- ja MID-tiedostoista. MIF-tiedosto sisältää tekstimuotoisena MID-tiedoston sisältämien paikkatietokohteiden kuvauksen. Vaikka MIF on suljettu formaatti, sen lukemiseen löytyy tuki esimerkiksi GDAL-kirjastosta ja useimmista paikkatietosovelluksista. (Open Source Geospatial Foundation, 2017b)

SHP on ESRI:n ArcGIS-tuoteperheen käyttämä suljettu vektoriformaatti. SHP-formaatissa paikkatietokohteista tallennetaan niiden geometria ilman topologiaa. Kuten muissakin vektoriformaateissa, kohteiden geometria tallennetaan pistejoukkoina. Formaatin käsittely erityisesti piirtämistarkoituksessa on nopeaa, sillä topologiatietoja ei tallenneta. Tiedostokoot ovat myös tästä johtuen useimmiten pienempiä kuin topologiatiedon tallentavilla formaateilla. (Environmental Systems Research Institute, 1998) GDAL-kirjasto tukee SHP-formaatin lukemista ja kirjoittamista.

Toteuttamani sovellus lataa Maastotietokannan karttalehtiä GML-formaatissa Kapsi ry:n palvelimelta. Karttalehdet ovat suorakulmaisia ja ne noudattavat UTM25-karttalehtijakoa. Yhden karttalehden koko on $24\,\mathrm{km} \times 12\,\mathrm{km} = 288\,\mathrm{km}^2$. Palvelimella jokainen UTM25-karttalehti on jaettu kahteen osaan puolittamalla se pystysuunnassa. Sovellus lataa asetustiedostossa määriteltyjen karttalehtien osat ja yhdistää ne kokonaiseksi kartaksi. Karttalehtien tiedostokoko vaihtelee kohteiden määrän mukaisesti. Käyttämieni karttalehtien tiedostokoko vaihteli 20 ja 100 megatavun välillä. Karttalehtien käsittelyyn vaadittavan muistin määrää rajoittaa se, että kohteista luetaan ja käsitellään GDAL-kirjaston avulla vain tietty osajoukko.

7.2.1. Maastotietokohteet

Maastotietokannan paikkatietokohteet eli maastotietokohteet on jaoteltu erilaisiin kohderyhmiin. Ryhmien jaottelulla on hierarkkinen rakenne. Kohteilla voi olla tietokannassa yhteyksiä toisiin kohteisiin, myös oman ryhmänsä ulkopuolelle. Kohderyhmiä on korkeimmalla tasolla 18. (Maanmittauslaitos, 2016) Jalkaisin liikuttaessa reitinhaun kannalta merkittäviksi kohderyhmiksi voidaan korkeimmalta tasolta laskea esimerkiksi *tiestö*, *maasto/1*, *maasto/2* ja *rakennukset*.



Kuva 17. Maastotietokannan maastotietokohteita sovelluksen visualisoimana.

Kuvassa 17 on ruudunkaappaus sovelluksen karttavisualisoinnista. Ruoveden läheisyyteen sijoittuvilta karttalehdiltä N4123 ja N4121 sovellus on ladannut konfiguroidut kohteet: järvet, rakennukset, suot, tieviivat, jyrkänteet, pellot, vesikuopat, kallioalueet, metsämaan kasvillisuuden ja virtavedet. Yhteensä näitä kohteita on miltei 60 000. Sovellukseen konfiguroidaan käytettävien kohteiden lisäksi, miten eri kohderyhmien paikkatietokohteet tulee piirtää. Suot on piirretty alueiksi vaalean sinisellä RGB-arvolla $(200, 230, 255)$, ja järvet on vastaavasti piirretty tummemman sinisellä RGB-arvolla $(0, 155, 255)$. Rakennukset ovat mustia alueita ja tieviivat on piirretty mustalla viivalla. Vaikka kohteita on paljon, koko kartan piirtäminen onnistuu suhteellisen nopeasti vieden keskimäärin 0,2 sekuntia. Sovellus piirtää koko kartan kerralla muistiin, jotta sitä voidaan liikutella nopeasti ruudulla.

Tiestö-kohderyhmän tieviiva kattaa reitinhaun kannalta kaikki merkittävät ih-

misen toimesta syntyneet kulkureitit. Tieviiva-kohteiksi on määritelty muun muassa tiet, kadut ja polut. Maastosta ei tallenneta ”heikkoja polkuja”, mutta käytännössä maastotietokanta kattaa kaikki tavanomaisilta maastokartoilta löytyvät polut. Tämä on tietenkin seurausta siitä, että maastokartat muodostetaan pääosin maastotietokannan aineiston pohjalta. (Maanmittauslaitos, 2016)

Tieviivoihin liittyy useita tarkentavia attribuutteja, jotka kuvaavat tieuran laatua ja mittoja. Tiestön geometria ja jotkin tiestön ominaisuustiedoista ovat osa laajempaa kansallista tietojärjestelmää, jonka hallinnointivastuu on Liikennevirastolla. Tieviivat tallennetaan keskiviivaperiaatteella, ja esimerkiksi risteyskohtien muodostuminen on kuvattu tarkasti kohdemallissa. Reitinhaun kannalta tieviivaa voidaan käsitellä sen korkeimmalla tasolla. Jalkaisin liikkuvan näkökulmasta kaikki tieviivaksi luokiteltavat kohteet ovat kuljettavissa. (Maanmittauslaitos, 2016)

Kohderyhmässä *maasto/1* on maanpinnan luontoon ja kasvillisuuteen liittyviä kohteita. Esimerkiksi suo on yksi tämän ryhmän kohteista. Suo tallennetaan alueena. Maastotietokohdemallissa on määritelty suolle vähimmäiskoko ja turvekerroksen minimipaksuus. Kohdemalli erottelee viljellyn suoalueen luonnontilaisesta suosta. Suoalueisiin liittyy myös suon kulkukelpoisuus- ja metsäisyystieto. Näiden attribuuttien arvot ovat kulkukelpoisuuden osalta helppokulkuinen ja vaikeakulkuinen, metsäisyyden ollessa joko metsää kasvava tai puuton. (Maanmittauslaitos, 2016)

Maasto/2-kohderyhmään kuuluvat vesistöihin liittyvät kohteet ja avoimet alueet. Esimerkiksi viivoina tallennettavat kosket kuuluvat tähän kohderyhmään. Kohdemallissa on määritelty, että viiva tallennetaan virtaveden juoksusuunnassa uoman keskelle. Kosken minimipituudeksi on määritelty 20 metriä. (Maanmittauslaitos, 2016)

Kohdemallin perusteella voidaan karkeasti arvioida, mitä kohderyhmiä reitinhaussa tulisi käyttää kussakin sovelluskohteessa. Kuten totesin, tarkka kohteiden rajaus vaatii merkittävästi selvitystyötä. Maanmittauslaitoksen kohdemalli on kuitenkin kattava ja täsmällinen, joten se soveltuu erinomaisesti selvitystyön tueksi. Kohdemallin lisäksi maastotietokohteiden tulkinnan apuna voidaan käyttää kohteiden XML-skeemoja, jotka ovat saatavilla Maanmittauslaitoksen palvelimelta. XML-skeemojen avulla on myös määritelty koodistot, joita maastotietokanta käyttää esimerkiksi soiden kulkukelpoisuus- ja metsäisyysluokkien koodauksessa.

7.2.2. Maastotietojen laatu

Maastotietokannan numeeristen kohteiden laatu muodostuu Maanmittauslaitoksen laatumallin määrittelyiden pohjalta. Malli määrittelee keinot laatutekijöiden mittaamiseksi ja asettaa vaatimuksia karttatuotteiden laadulle. Maastotietokannan kohteita kerätään useista eri lähteistä. Arvioimalla niitä laatumallin avulla tietokanta pysyy korkealaatuisena. Laatumalli määrittelee koordinaattitietojen laadun sijaintikeskivirheenä, joka lasketaan poikkeamien neliökeskiarvona. Maastotietokannan kohteiden keskivirheitä seurataan jatkuvasti laadun takaamiseksi. Laatumalli huomauttaa myös, että koordinaattipistevälin tulee olla geometrisesti jatkuvien kohteiden yhteydessä mahdollisimman suuri. Tällöin pisteiden määrä on mahdollisimman pieni, ja tarpeettomalta laskennalta vältytään. (LAJA-työryhmä, 1995)

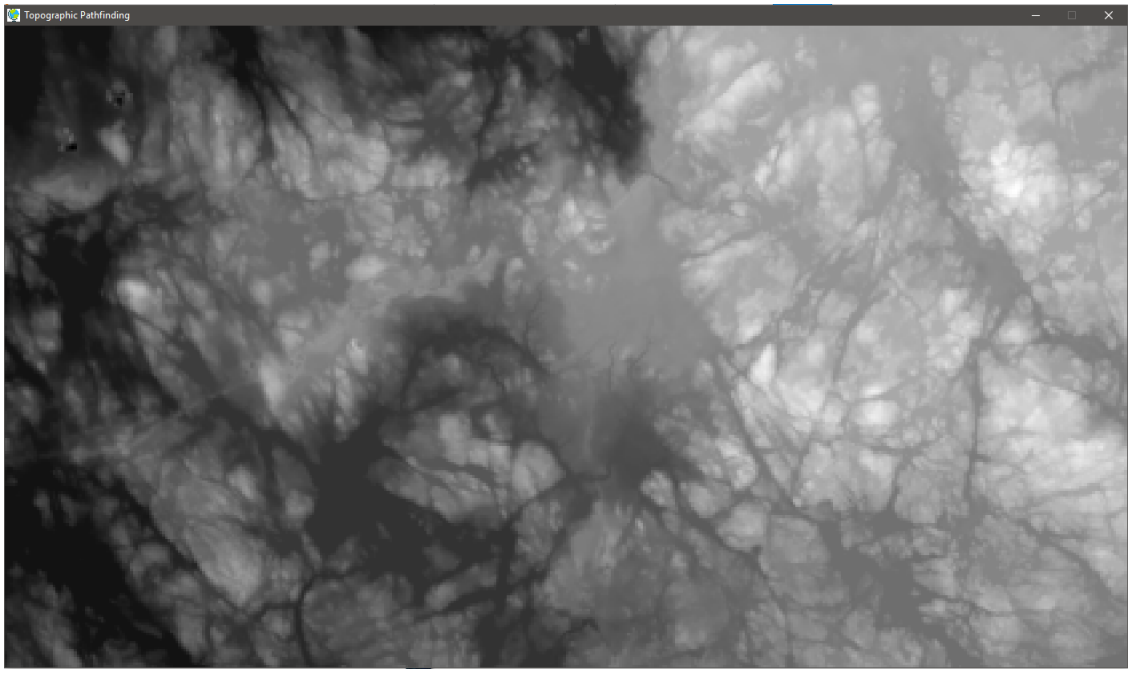
Paikkatietokohteiden osalta laatumalli asettaa vaatimukset esimerkiksi sijaintitarkkuudelle, ajantasaisuudelle ja kattavuudelle. Sijaintitieto on jaettu kahteen eri laatuluokkaan, A ja B. Näissä laatuluokissa on erikseen määritellyt sijaintitarkkuudet metreinä kullekin kohdetyypille. Sijaintitarkkuudella tarkoitetaan siis sijaintikeskivirhettä. Sijaintitietojen tarkkuus vaihtelee kohteen mukaisesti. Yhteiskunnan tarpeisiin rakennettujen kohteiden sijainnit tunnetaan tarkasti, joten niiden laadulle asetetaan korkeammat vaatimukset. Sijaintitarkkuus suurjännitelinjan pylväälle on laatuluokassa A kolme metriä. Uusien rakennusten sijaintitarkkuus on maastotietokannassa kaksi metriä. Laajojen luonnontilaisten alueiden, kuten esimerkiksi soiden, sijaintitiedot eivät ole niin tarkkoja. Sijaintitarkkuudet vaihtelevat kahdesta metristä 80 metriin.

Sijaintitarkkuuksista voidaan arvioida reitinhakuun soveltuvaa ruudukointikokoa. Vaihtelevat sijaintitarkkuudet vaikuttavat vähemmän korkeammalla abstraktiotasolla, eli tässä tapauksessa kartan päälle muodostetulla ruudukolla. Pienet poikkeamat sijaintitiedoissa eivät vielä välttämättä siirrä kohdetta viereiseen ruutuun, jolloin rasteroinnin tulos tulos ei muutu. Sijaintitarkkuudet löytyvät myös maastotietokannan GML-karttalehdistä. Jokaiseen paikkatietokohteeseen liittyy tieto sijainti- ja korkeustarkkuudesta. Nämä tiedot ovat helposti luettavissa GDAL:n avulla, joten niiden integroiminen reitinhakuun on ainakin teoriassa mahdollista. Aineiston epätarkkuudesta ei voi päästä eroon, mutta epätarkkuuden aiheuttaman poikkeaman tilastoiminen voi auttaa ymmärtämään saatuja tuloksia. Joka tapauksessa maastotietokanta on ehdottomasti tarkkuudeltaan paras saatavilla oleva aineisto. Sen tarkkuus on täysin riittävä esimerkiksi jalkaisin tai moottoriajoneuvolla maastossa kuljettavan reitin hakuun.

7.3. Korkeusmallin rasterointi

Maanmittauslaitos tarjoaa korkeusmallin ladattavaksi sekä 10 m × 10 m että 2 m × 2 m ruudukkokoolla. Näistä käytetään myös nimityksiä KM10 ja KM2. Suuremmalla ruudukkokoolla korkeustiedon tarkkuus on 1,4 metriä. Korkeusmallit ilmaisevat korkeustiedon metreinä siten, että merenpinnan alapuoliset kohteet ovat negatiivisina arvoina. (Maanmittauslaitos, 2017e) Maanmittauslaitoksen KM2 perustuu laserkeilaukseen, ja KM10 on muodostettu korkeuskäyristä sekä maastotietokannan paikkatietokohteiden korkeuksista. (Paikkatietokeskus, 2017)

Korkeusmallin osalta saatavilla olevina tiedostomuotoina on GeoTIFF, PNG ja pelkkä teksti. Käyttämästäni GDAL-kirjaston versiosta puuttui tuki GeoTIFF- ja PNG- formaattien käsittelyyn taulukoina, joten sovellus lataa ja lukee 10 m × 10 m ruudukkokoon korkeusmallin tekstimuodossa. Tekstimuotoisen korkeusmallin rivit koostuvat x- ja y-koordinaattien avulla määritellyistä pisteistä ja näiden pisteiden korkeusarvoista. Jokainen karttalehti on jaettu kahdeksaan osaan. Yhden 24 km × 12 km karttalehden kattava korkeusmalli on kooltaan yhteensä 88,3 megatavua. Sovellus rasteroi korkeusmallin vastaamaan käytettyä ruudukkoa, joten muistissa pidettävän korkeusmallin koko on huomattavasti pienempi. Tarkasta 10 m × 10 m korkeusmallista interpoloidaan suuremmalle rasterille korkeustietoja.



Kuva 18. Karttalehden M4224 normalisoidun korkeusmallin visualisointi sovelluksessa.

Kuvassa 18 on ruudunkaappaus sovellukseni esittämästä karttalehden M4224 korkeusmallista. Kartalla näkyvän alueen korkeusmallin arvot on normalisoitu välille $[0, 255]$, jotta niitä voidaan käyttää pikselien harmaasävyarvoina. Korkeat alueet näkyvät siis mallissa vaaleina ja merenpintaan nähden matalammat alueet tummina. Sovellus kykenee hakemaan reittejä myös pelkän korkeusmallin perusteella. Tämä on erityisen hyödyllistä korkeusmallin kustannusalgoritmia kehittäessä.

Korkeusmalli huomioidaan reitinlaskennassa ruutujen välillä etenemisen kustannuksena. Siinä missä yleisesti paikkatietokohteista lasketut kustannukset eivät riipu kulkusuunnasta, korkeusmallin osalta kulkusuunta voi vaikuttaa kustannukseen. Reitin optimaalisuutta määriteltäessä voidaan esimerkiksi todeta, että alamäkeen on kevyempi kulkea. Sovelluksen reitinhakualgoritmi tarkistaa korkeusmallissa määritellyn kustannuksen siten, että suunta on huomioitu. Tämä mahdollistaa esimerkiksi algoritmit jotka välttävät jyrkkiä nousuja nostamalla niiden kustannusta eksponentiaalisesti, mutta toisaalta sallivat etenemisen alaspäin samaista rinnettä.

7.4. Maastotietokohteiden rasterointi

Kuten kohdassa 3.2 mainitsin, paikkatietokohteita tallennetaan joko viivoina, alueina tai pisteinä. Tämä pätee myös maastotietokantaan. Maastotietokannan kohdemalli määrittelee, miten paikkatieto muodostetaan ja tallennetaan kunkin kohteen osalta. Esimerkiksi suot tallennetaan alueena ja metsämaan kasvillisuus pisteenä. Tämä asettaa vaatimuksen laajalle valikoimalle rasterointialgoritmeja. Rasterointialgoritmien ideana on muodostaa yksittäisen ruudun geometriatiedon ja paikkatietokohteiden perusteella jokin yksittäinen numeerinen arvo. Sovelluksessa nämä algoritmit on vielä jaettu kahteen osaan, rasterointiin ja jälkiprosessointiin.

Rasterointi on laskennallisesti erittäin raskas prosessi. Esimerkiksi ruudukkokoolla $50\,\mathrm{m} \times 50\,\mathrm{m}$ yksi $228\,\mathrm{km}^2$ karttalehti koostuu 115 200 ruudusta. Jokaiselle ruudulle tulee laskea arvo kaikkien valittujen paikkatietokohdeluokkien osalta. Laskennan raskaus johtuu spatiaalisesta ulottuvuudesta ja paikkatietokohteiden suuresta määrästä. Mikäli jokaisen ruudun kohdalla joudutaan tutkimaan kaikki kartan alueella olevat paikkatietokohteet rasteriarvon muodostamiseksi, rasteroinnin kompleksisuus kasvaa kohtuuttoman suureksi. Geometriset operaatiot, kuten esimerkiksi leikkaus, vaativat paljon laskentaa erityisesti kohteiden ollessa geometrioiltaan monimutkaisia. Rasterointia voidaan kuitenkin tehostaa käyttämällä jotakin paikkatietoindeksiä ruudun alueella tai sen läheisyydessä olevien paikkatietokohteiden hakemiseen.

Toteuttamani sovellus muodostaa kohdeluokkakohtaiset paikkatietoindeksit käyt-

tämällä kohdassa 3.3 käsittelemääni R-puuta. Näin esimerkiksi kaikki suoalueet löytyvät samasta indeksistä. Suokohteita rasteroidessa kunkin ruudun tapauksessa indeksin avulla saadaan nopeasti haettua ne suot, jotka voivat vaikuttaa ruudun rasteriarvoon. R-puuta käytetään yhtä lailla pisteiden, viivojen ja alueiden tallentamiseen. Lisäksi rasteroinnin tehostamiseksi sovellus voidaan konfiguroida yksinkertaistamaan paikkatietokohteiden geometriaa ennen rasterointia. Tähän sovellus käyttää GDAL-kirjaston tarjoamaa algoritmia, joka perustuu Douglasin ja Peuckerin (1973) iteratiiviseen algoritmiin. Algoritmille määritellään kohdeluokkakohtainen toleranssi. Yksinkertaistetun geometrian pisteet ovat enintään toleranssin päässä alkuperäiseen geometriaan nähden.

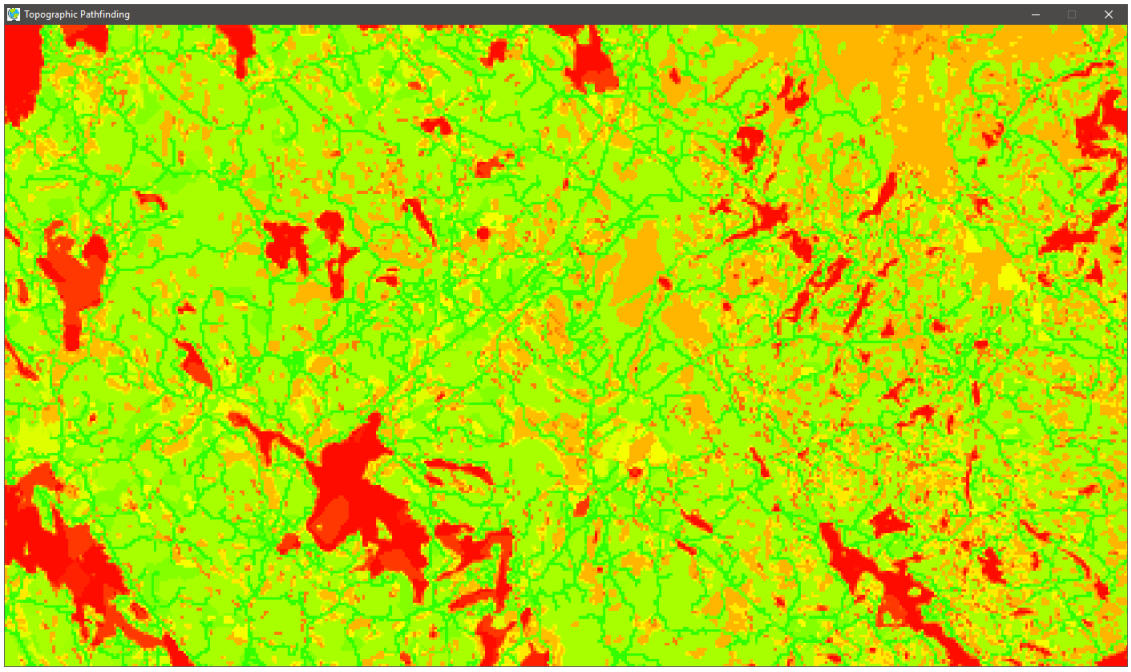
Sovelluksessa käyttämäni Rtree Python-moduuli kykenee luomaan indeksejä rajaavien suorakaiteiden perusteella, joten paikkatietokohteiden geometriasta indeksoidaan vain näiden rajaavat suorakaiteet. Tällöin rasterointialgoritmeille voi välittyä myös esimerkiksi sellaisia viivakohteita, jotka eivät missään kohtaa kulje rasteroitavan ruudun alueella. Rasterointialgoritmin tarvitsee siis vielä erikseen suodattaa indeksin täsmäämät kohteet, mutta tämä on laskennallisesti todella kevyttä verrattuna kaikkien kohteiden käsittelyyn ilman indeksiä. Toisaalta esimerkiksi pistekohteiden osalta indeksi toimii täydellisesti täsmäten vain ne kohteet, jotka todella sijoittuvat ruudun sisälle.

R-puun avulla voidaan myös etsiä k lähintä naapuria ruudulle paikkatietokohteiden joukosta. Tämä on erityisen hyödyllistä esimerkiksi jotakin alueellista ominaisuutta kuvaavien epäsäännöllisten pistetietojen tapauksessa. Maastotietokannassa esimerkiksi metsämaan kasvillisuus on tallennettu epäsäännöllisinä pisteinä noin 300 metrin välein. Pienellä ruudukkokoolla kaikkien ruutujen sisälle ei osu tällaista kasvillisuuspistettä, joten ruudun kasvillisuustieto tulee johtaa k:n lähimmän naapurin perusteella.

Yksinkertaisimmillaan rasterointialgoritmi voi esimerkiksi vain tarkistaa, osuuko tietyn kohdeluokan olio ruudun alueelle. Toisaalta esimerkiksi alueiden tapauksessa voidaan tarkastella pinta-alan ja paikkatietokohteiden pinta-alan suhdetta. Pistekohteiden osalta voidaan esimerkiksi laskea ruudun sisälle jäävien kohteiden määrä. Sovelluksessa on mahdollista määritellä kullekin maastotietokohdeluokalle oma rasterointialgoritmi. Rasteroinnin tulokset tallennetaan levylle olioina, joten rasterointialgoritmit voivat tuottaa myös äärimmäisen monimutkaisia tietotyyppejä pelkkien numeeristen arvojen lisäksi. Rasterointialgoritmi voi vaikkapa tallentaa ruudun k lähimpien naapurien tietyn attribuutin arvojen frekvenssit.

Rasteroinnin tuloksista sovellus jälkiprosessoi kustannusrasterin konfiguraation

perusteella. Tätä kustannuskonfiguraatiota voidaan vapaasti muokata ilman tarvetta uudelleenrasteroinnille. Kustannuskonfiguraatio määrittelee jälkiprosessointialgoritmit kohdeluokittain. Jälkiprosessointialgoritmit muuttavat rasterointialgoritmien tulosoliot kustannusta kuvaaviksi liukulukuarvoiksi. Lisäksi konfiguraatiossa on määritelty, kuinka eri kohdeluokkia tulee painottaa. Lopullinen kustannusrasteri muodostuu summaamalla eri kohdeluokkien rasteriarvot yhteen painotukset huomioiden. Kohdeluokkien konfigurointi mahdollistaa myös negatiivisten arvojen käyttämisen. Jos kokonaiskustannus menee jonkin ruudun osalta negatiiviseksi, käytetään kustannuksena tämän ruudun osalta nollaa. Esimerkiksi tieviivojen yhteydessä voi olla hyödyllistä käyttää negatiivisia arvoja, sillä ruudulla kulkevan tien voidaan katsoa laskevan ruudun kustannusta.



Kuva 19. Sovelluksen muodostaman kustannusrasterin visualisointi.

Kuvassa 19 on ruudunkaappaus sovelluksen muodostaman kustannusrasterin visualisoinnista karttalehdellä M4224. Arvoja on visualisointia varten normalisoitu sopivalle välille, jotta kustannukset muodostavat liukuvärin vihreästä punaiseen. Punainen kuvaa tässä visualisoinnissa korkeinta kustannusta. Tieviivat erottuvat tästä rasterista vielä heikosti vihreinä viivoina, jotka halkovat korkeampikustanteisia alueita. Selkeimpinä esteinä ovat järvet, jotka värjäytyvät täysin punaiseksi. Erilaiset yhdistelmät kasvillisuustyyppejä ja suoalueita ovat nähtävissä keltaisen ja oranssin sävyinä. Ihmissilmä ei kykene erottelemaan tästä visualisoinnista kaikkein hienovaraisimpia vaihteluja kustannuksissa, mutta siitä saa hyvän yleiskuvan alueen

keskeisistä esteistä ja kuljettavista alueista.

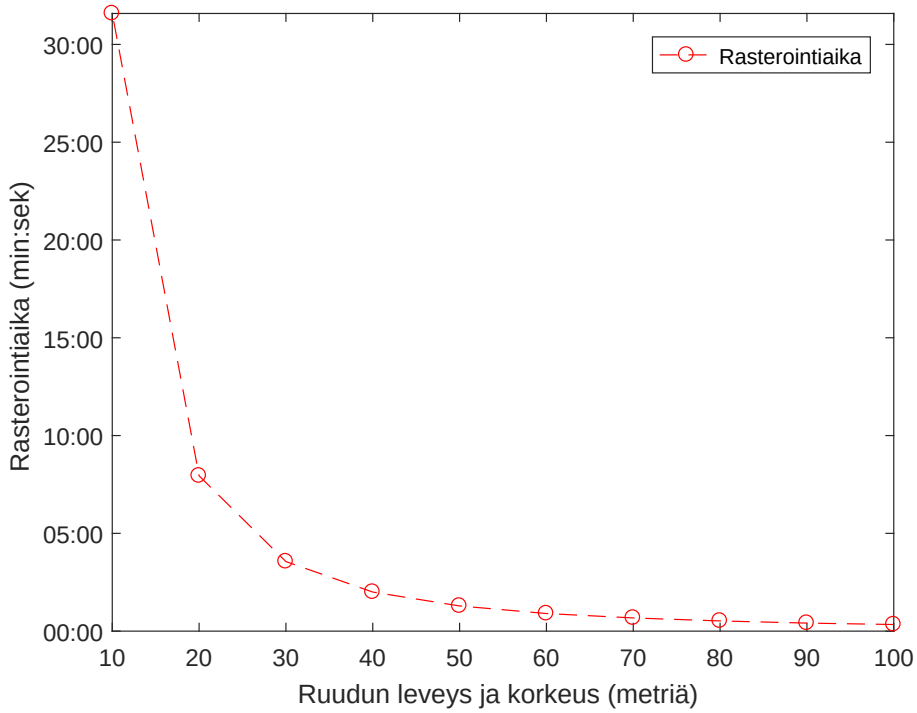
7.5. Sovelluksen suorituskyky

Käyttämäni menetelmä tuottaa hyvälaatuisia polkuja kustannuskonfiguraation puitteissa. Esimerkiksi jalkaisin liikkuvalle muodostamani kustannuskonfiguraation tapauksessa reitit etenevät mahdollisimman paljon polkuja ja teitä pitkin, kiertäen suot ja järvet lyhyintä mahdollista reittiä. Maastokartoilla esteitä muodostuu kartalle huomattava määrä, mutta reitinhaun suoritusajat ovat silti kohtuullisia. Korkeusmallin rasterointi on kustannusrasterin muodostamiseen nähden laskennallisesti kevyt prosessi. Kustannusrasterin laskenta esimerkiksi 288 km^2 alueelle ruudukkokoolla 50 m $\times$ 50 m kestää joitakin minuutteja, riippuen kohteiden määrästä. Etukäteen tallennetun rasterin jälkiprosessointi kestää joitakin sekunteja.

Sovellus merkitsee lyhyimmän polun käyttäen ruudukon ruutujen keskipisteitä. Ruudukoinnin aiheuttama poikkeama reitteihin pienenee hyväksyttävälle tasolle käytettäessä ruudukkokokoa 50 m $\times$ 50 m tai tätä pienempää. Tällöin reitit seuraavat esimerkiksi polkuja riittävän täsmällisesti myös suuremmalla mittakaavalla. Toki pienelläkin ruudukkokoolla rasterointi voi aiheuttaa kohdassa 3.4 kuvaamani kaltaisia ongelmia, joissa rasterin tarkkuus ei riitä kuvaamaan kartalla olevaa selkeää estettä. Tältä voidaan kuitenkin välttyä määrittelemällä rasterointialgoritmit riittävän tiukoiksi esteinformaation hukkaamisen suhteen. Yhtä sovelluksen algoritmeista voidaan käyttää rasteriarvon määrittämiseksi pelkästään sen perusteella, leikkaako kyseinen ruutu yhtäkään tietyn kohdeluokan paikkatietokohdetta.

Kaaviossa 1 on Ilomantsiin sijoittuvan karttalehden P5333 rasterointiajat. Pienin ruudukkokoko on 10 m $\times$ 10 m, jonka rasterointi kesti 31 minuuttia ja 34 sekuntia. Rasterointi kohdistui kymmeneen eri kohdeluokkaan: järvi, rakennus, suo, tieviiva, jyrkänne, metsämaan kasvillisuus, vesikuoppa, virtavesi, maatalousmaa ja kallioalue. Odotetun kaltaisesti rasterointiaika vähenee eksponentiaalisesti ruudukkokoon kasvaessa. Ruudukkokoolla 50 m $\times$ 50 m kartan rasterointi kesti reilun minuutin. Tämä on jo hyvinkin kohtuullinen esiprosessointiaika ottaen huomioon, että sen tulokset voidaan tallentaa levylle.

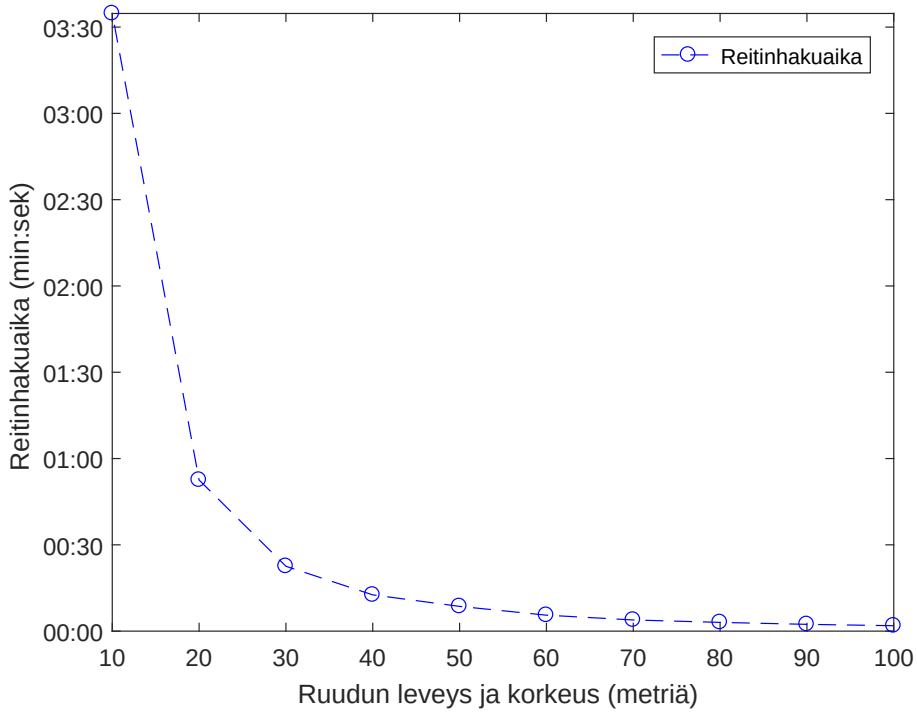
On tärkeää huomata, että sovelluksen tallentaman rasterointituloksen koko levyllä riippuu käytetystä ruudukointikoosta ja rasterointialgoritmien tulosolioiden tavukoosta. Karttalehden kohteiden määrä ei vaikuta tallennettavan rasterin kokoon. Valittujen kohdeluokkien määrä sen sijaan vaikuttaa, sillä rasterointitulokset on tallennettava kohdeluokittain. Esimerkiksi karttalehden P5333 tapauksessa levylle tal-



Kaavio 1: Karttalehden P5333 rasterointiajat eri ruudukon leveyksillä.

lennetun rasterointituloksen koko on 138 megatavua käytettäessä ruudukkokokoa 10 m × 10 m ja edellä mainittuja kohdeluokkia. Tällä ruudukkokoolla ja näillä kohdeluokilla Suomen maa-alueiden rasterointituloksien koko olisi noin 158 gigatavua. Käytettäessä ruudukkokokoa 50 m × 50 m rasterointituloksen koko levyllä oli vain noin 5,7 megatavua. Tällä ruudukkokoolla Suomen maa-alueet voitaisiin rasteroida noin 6,5 gigatavun levypinnalle. Tiedostokoot kasvavat eksponentiaalisesti rasterointiajan tapaan ruudukointikoon vaikuttaessa käsiteltävien ruutujen määrään.

Kaaviossa 2 on kuvattu reitinhaun kesto eri ruudun leveyksillä. Käytetty lähtöpiste on karttalehden P5333 koillisnurkassa ja maalipiste on samaisen karttalehden lounaisnurkassa. Pisteiden välinen euklidinen etäisyys on siis noin 26,8 kilometriä. Löytyvän optimaalisen reitin pituus vaihtelee käytetyn ruudukointikoon mukaan, sillä ruudukkokoko vaikuttaa siihen, kuinka tarkkoja yksityiskohtia rasteri pystyy mallintamaan. Esimerkiksi 10 m × 10 m ruudukolla optimaalinen reitti on 38,3 kilometriä pitkä, kun taas 50 m × 50 m ruudukolla reitin pituus on 37,9 kilometriä. Suurimmalla testatulla koolla (100 m × 100 m) optimaalinen reitti on 39,6 kilometriä pitkä. Kuvassa 20 on esitetty reittien eroavaisuus P5333 karttalehdellä kolmella eri ruudukkokoolla.

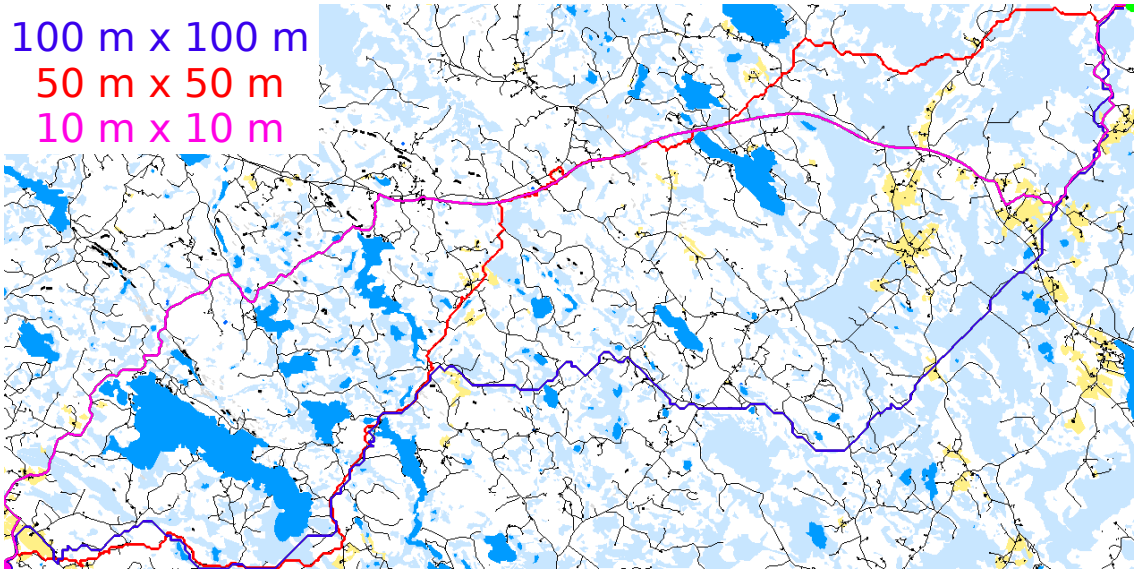


Kaavio 2: Karttalehden P5333 reitinhakuajat eri ruudukon leveyksillä.

Myös reitin laskemiseen kuluva aika riippuu odotetusti ruudukointikoosta. Tiheämmillä ruudukoilla A*-algoritmin näkemä hakuavaruus on suurempi, jolloin reitin löytämiseksi on käsiteltävä väistämättä suurempi määrä ruutuja. Tietyn reitin hakemiseen kuluva aika ei juuri vaihtele reitinhaun toimiessa täysin deterministisesti. Käsiteltyjen solmujen määrä on siis sama eri suorituskerroilla.

Ruudukkokoon pienentyessä myös ajonaikaisen muistin tarve kasvaa. Ruudukkokoolla 10 m × 10 m sovellus käytti parhaimmillaan liki 2,2 gigatavua muistia. Toisaalta myös reitin hakeminen kesti yli kolme minuuttia. Kyseinen ruudukkokoko ei siis välttämättä ole mielekäs näin pitkiä reittejä hakiessa. Käytettäessä 50 m × 50 m ruudukkokokoa reitinhaku kesti keskimäärin 8,6 sekuntia. Hakuun kuluvaan aikaan vaikuttaa ensisijaisesti optimaalisen reitin pituus, joka taasen määräytyy ruudukkokoon, kustannuskonfiguraation ja karttalehden paikkatietokohteiden perusteella.

Reitinhaun suorituskyvyn riippuessa merkittävästi käytetyistä rasterointialgoritmeista ja kustannuskonfiguraatiosta näiden testitulosten perusteella ei voida vielä tehdä tarkkoja päätelmiä menetelmän tehokkuudesta ja reittien todellisesta laadusta. Kuten kohdassa 5.2 mainitsin, A*-algoritmi on erityisen herkkä kustannuksen ja heuristiikan tasapainon ja mittakaavan suhteen. Ilman tarkkaa kuljettavuustutki-



Kuva 20. Eri ruudukoilla optimaaliset reitit poikkeavat toisistaan.

musta tätä tasapainoa ei voida saavuttaa, sillä kustannuksen ja heuristiikan arvot eivät ole samassa yksikössä. Reitinhaku toimi kuitenkin suhteellisen ripeästi jo karkean arvioni perusteella muodostetuilla kustannus- ja rasterointikonfiguraatioilla, jotka käyttivät jopa kymmentä eri maastotietokannan kohdeluokkaa.

8 Yhteenveto

Reitinhakua maastokartoilla voidaan lähestyä kahdella eri tavalla. Esittelemäni ja toteuttamani rasterointimenetelmän selkeä etu on sen yksinkertaisuus. Rasterin ja graafien välinen yhteys tekee reitinhakualgoritmien soveltamisesta suoraviivaista, joskin tehokkuuden kannalta huomioitavia seikkoja on useita. Menetelmän laskennalliset kustannukset ovat myös hyvin ennustettavissa, sillä ne riippuvat käytetystä ruudukointikoosta. Toisaalta ruudukointikoko vaikuttaa merkittävästi siihen, kuinka tarkasti maastokartan yksityiskohtia voidaan mallintaa. Kuten kohdassa 7.5 mainitsin, optimaaliset reitit poikkeavat toisistaan paljonkin ruudukkokoon muuttuessa.

Maastokartat ovat paikkatietoaineistoina monipuolisia. Ne sisältävät määrällisesti paljon kohteita, ja lisäksi erilaisia kohdeluokkia on paljon. Reitinhakua toteuttava paikkatietojärjestelmä joutuu siis väistämättä käsittelemään suuria määriä dataa. Paikkatietoindeksit ovat välttämätön tietorakenne miltei minkä tahansa paikkatietojärjestelmän yhteydessä. Myös reitinhaussa tulee hyödyntää paikkatietoindeksejä järjellisten laskenta-aikojen saavuttamiseksi. Rasterointimenetelmä yksinkertaistaa vektoriaineistoa, mutta itse rasterointiprosessi voi olla hyvinkin monimutkainen. Kuten totesin, rasteroinnissa on mahdollistettava laaja valikoima erilaisia menetelmiä paikkatietokohteiden käsittelyyn. Vaikka käsiteltäviä geometrioita on vain piste, viiva ja alue, näillä geometrioilla voidaan kuvata reitinhaun kannalta hyvinkin monenlaisia kohteita.

Käyttämäni A*-algoritmi on erinomainen valinta reitinhakuun ruudukoilla, mutta oikeiden parametrien valinta on ensisijaisen tärkeää optimaalisten reittien löytämiseksi. Reitinhaku tarvitsee tarkan määrittelyn optimaalisuudelle. Käyttämällä numeerista, yksikötöntä kustannusta reitinhaku saadaan toimimaan kohtalaisesti. Sopivien parametrien haku on tiettyyn pisteeseen asti suhteellisen yksinkertaista. Jos reitinhaku tekee esimerkiksi tarpeettoman innokkaasti vesistönylityksiä reitin varrella, voidaan vesistöjen kustannusta nostaa. Vastaavasti voidaan toimia muiden paikkatietokohteiden kanssa. Mikäli reitinhaku ei hyödynnä polkuja riittävän tehokkaasti, voidaan polkujen aiheuttamaa ruudun kustannuksen vähenemistä korostaa.

Kustannusten säätämisen keskeinen ongelma on se, että kustannukset eivät ole samalla mittakaavalla heuristiikkaan nähden. Tällöin reitinhaku löytää kyllä kustannuksiin nähden optimaalisen polun, mutta reitinhaku kestää tarpeettoman pitkään. Huolellisesti muodostettu kustannusmalli maksimoi heuristiikalla saavutettavan hyödyn, jolloin reitinhaku etenee maaliin mahdollisimman suoraan optimaalista reittiä noudattaen.

Vaikka toteuttamani sovelluksen suorituskyky onkin kohtuullinen, esimerkiksi esiprosessointia voitaisiin tehostaa entisestään hyödyntämällä esimerkiksi moniytimellisyyttä. Rasteroinnin suhteen tuloksia voitaisiin pakata, jotta ne veisivät vähemmän tilaa levyllä. Esimerkiksi kohdassa 7.5 mainitsemani 137 megatavun kokoinen rasteroinnin tulostiedosto kutistuu noin 3,7 megatavuun LZMA-pakkauksella. Tehotyöasemalla tarkkojen, kymmeniäkin kilometrejä pitkien reittien laskenta onnistuu sekunneissa, mutta mobiilikäyttöön tällainen reitinlaskenta soveltuu huonosti. Kuten luvussa 7 mainitsin, reitinhaku voitaisiin ulkoistaa myös erilliselle palvelimelle. Tällöin pienempitehoisetkin päätelaitteet voivat hyötyä reitinhausta, eikä laskentaa tarvitse tehdä etukäteen erillisellä työasemalla.

Laskennallisen geometrian ja prosessoriteknologian kehittyessä painotettuja alueita käsittelevät menetelmät ovat entistä houkuttelevampia reitinhakuun maastokartoilla. Vaikka ne ovat laskennallisesti monimutkaisia, kyky tuottaa täsmällisiä tuloksia antaa niille huomattavan etulyöntiaseman. Rasteroinnin aiheuttama yksityiskohtien häviäminen saattaa poistaa esimerkiksi jalkaisin liikkuvan kannalta joitakin äärimmäisen oleellisia maastonkohtia. Esimerkiksi jonkin vesistön ylittävä kapea kannas saattaa hukkua rasteroinnin seurauksena, jolloin reitinhakualgoritmin löytämä reitti on huomattavasti pidempi kuin todellinen lyhyin reitti. Rasterimenetelmää käytettäessä joudutaan usein tekemään kompromisseja, joskin joidenkin paikkatietokohteiden käsittelyyn se sopii mainiosti.

Viitteet

Antikainen, H. (2013). Using the hierarchical pathfinding A* algorithm in GIS to find paths through rasters with nonuniform traversal cost. *ISPRS International Journal of Geo-Information*, *2*(4), 996–1014.

Aref, W. G., & Samet, H. (1991). Extending a DBMS with spatial operations. In *Proceedings of the Second International Symposium on Advances in Spatial Databases* (pp. 299–318).

Balstrøm, T. (2002). On identifying the most time-saving walking route in a trackless mountainous terrain. *Danish Journal of Geography*, *102*(1), 51–58.

Bjornsson, Y., Enzenberger, M., Holte, R., Schaejfer, J., & Yap, P. (2003). Comparison of different grid abstractions for pathfinding on maps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (pp. 1511–1512). Morgan Kaufmann.

Botea, A., Müller, M., & Schaeffer, J. (2004). Near optimal hierarchical pathfinding. *Journal of Game Development*, *1*(1), 7–28.

Bugayevskiy, L. M., & Snyder, J. (1995). *Map Projections: A Reference Manual*. CRC Press.

Chaplot, V., Darboux, F., Bourennane, H., Leguédois, S., Silvera, N., & Phachomphon, K. (2006). Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density. *Geomorphology*, *77*(1), 126–141.

Cui, X., & Shi, H. (2012). An overview of pathfinding in navigation mesh. *International Journal of Computer Science and Network Security*, *12*(12), 48–51.

Dechter, R., & Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, *32*(3), 505–536.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numererische Mathematik*, *1*(1), 269–271.

Douglas, D. H. (1994). Least-cost path in GIS using an accumulated cost surface and slopelines. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *31*(3), 37–51.

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122.

Endomondo. (2017). *Endomondo.* Retrieved 2017-01-27, from `https://www.endomondo.com`

Environmental Systems Research Institute. (1998). *ESRI Shapefile Technical Description.* Retrieved 2017-05-10, from `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`

Environmental Systems Research Institute. (2015). *Independent report highlights ESRI as leader in global GIS market* [Press release]. Retrieved from `http://www.esri.com/esri-news/releases/15-1qtr/independent-report-highlights-esri-as-leader-in-global-gis-market`

Environmental Systems Research Institute. (2017). *ArcGIS pro.* Retrieved 2017-02-28, from `http://pro.arcgis.com/en/pro-app/`

Finkel, R. A., & Bentley, J. L. (1974). Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, *4*, 1–9.

Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, *34*(3), 596–615.

Galati, S. R. (2006). *Geographic Information Systems Demystified.* Artech House.

Gatrell, A. C. (1991). Concepts of space and geographical data. In *Geographical Information Systems: Principles and Applications* (pp. 119–134). Longman.

Goodchild, M. F. (2009a). Geographic information system. In *Encyclopedia of Database Systems* (pp. 1231–1236). Springer.

Goodchild, M. F. (2009b). Geographic information systems and science: Today and tomorrow. *Annals of GIS*, *15*(1), 3–9.

GRASS development team. (2013). *Applications.* Retrieved 2017-02-28, from `https://grass.osgeo.org/documentation/applications/`

GRASS development team. (2015). *General overview.* Retrieved 2017-02-28, from `https://grass.osgeo.org/documentation/general-overview`

GRASS development team. (2017). *Historical notes.* Retrieved 2017-02-28, from https://grass.osgeo.org/home/history/

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data* (pp. 47–57). ACM.

gvSIG association. (2017). *Introduction.* Retrieved 2017-02-28, from http://downloads.gvsig.org/download/web/en/build/html/user_manual/2.3/manual/003.html

Hale, D. H. (2011). *A growth-based approach to the automatic generation of navigation meshes* (PhD thesis). The University of North Carolina at Charlotte.

Harabor, D. D., & Grastien, A. (2013). An optimal any-angle pathfinding algorithm. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling* (pp. 308–311). AAAI.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, *4*(2), 100-107.

Helft, M. (2016). *The Godfather of Digital Maps.* Retrieved 2017-02-28, from https://www.forbes.com/sites/miguelhelft/2016/02/10/the-godfather-of-digital-maps

Henrich, A., Six, H., & Widmayer, P. (1989). The LSD tree: Spatial access to multidimensional point and nonpoint objects. In *Proceedings of the Fifteenth International Conference on Very Large Databases* (pp. 45–53).

Hertel, S., & Mehlhorn, K. (1985). Fast triangulation of the plane with respect to simple polygons. *Information and Control*, *64*(1-3), 52–76.

Hexagon Geospatial. (2014). *Easily Unlocking and Utilizing Geodatabases.* Retrieved 2017-03-01, from http://www.hexagongeospatial.com/white-papers/easily-unlocking-and-utilizing-geodatabases

Hexagon Geospatial. (2017). *GeoMedia.* Retrieved 2017-03-01, from http://www.hexagongeospatial.com/products/power-portfolio/geomedia

Heywood, I., Cornelius, S., & Carver, S. (2006). *An Introduction to Geographical Information Systems* (3rd ed.). Pearson Education.

Julkisen hallinnon tietohallinnon neuvottelukunta. (2016a). *JHS 196 EUREF-FIN -järjestelmän mukaiset koordinaatit Suomessa.* Retrieved 2017-4-9, from http://docs.jhs-suositukset.fi/jhs-suositukset/JHS196/JHS196.pdf

Julkisen hallinnon tietohallinnon neuvottelukunta. (2016b). *JHS 197 EUREF-FIN -koordinaattijärjestelmät, niihin liittyvät muunnokset ja karttalehtijako.* Retrieved 2017-4-9, from http://www.jhs-suositukset.fi/suomi/jhs197

Kapsi Internet-käyttäjät ry. (2017). *Kartat.kapsi.fi.* Retrieved 2017-05-09, from http://kartat.kapsi.fi

Karjalainen, S. (2017). *Index of /topopath.* Retrieved 2017-5-19, from http://virtuoosi.kapsi.fi/topopath/

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109.

Korf, R. E., Reid, M., & Edelkamp, S. (2001). Time complexity of iterative-deepening-A*. *Artificial Intelligence*, *129*(1-2), 199–218.

LAJA-työryhmä. (1995). *Maastotietojen laatumalli.* Retrieved 2017-05-10, from http://maanmittauslaitos.fi/sites/maanmittauslaitos.fi/files/old/Maastotietojen_laatumalli.pdf

Longley, P., Goodchild, M., Maguire, D., & Rhind, D. (2005). *Geographic Information Systems and Science* (2nd ed.). John Wiley & Sons.

Maanmittauslaitos. (2016). *Maanmittauslaitoksen maastotietokohteet.* Retrieved 2017-5-9, from http://maanmittauslaitos.fi/sites/maanmittauslaitos.fi/files/old/maastotietokohteet.pdf

Maanmittauslaitos. (2017a). *Avoimien aineistojen tiedostopalvelu.* Retrieved 2017-04-11, from http://www.maanmittauslaitos.fi/asioi-verkossa/avoimien-aineistojen-tiedostopalvelu

Maanmittauslaitos. (2017b). *Karttapaikka.* Retrieved 2017-5-15, from https://asiointi.maanmittauslaitos.fi/karttapaikka/

Maanmittauslaitos. (2017c). *Maanmittauslaitoksen avoimen tietoaineiston CC 4.0 -lisenssi.* Retrieved 2017-05-09, from http://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/asiantuntevalle-kayttajalle/maastotiedot-ja-niiden-hankinta/avoimen

Maanmittauslaitos. (2017d). *Maastotietokanta.* Retrieved 2017-05-09, from `http://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/asiantuntevalle-kayttajalle/tuotekuvaukset/maastotietokanta`

Maanmittauslaitos. (2017e). *Tuotekuvaukset.* Retrieved 2017-05-10, from `http://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/asiantuntevalle-kayttajalle/tuotekuvaukset`

Manolopoulos, Y., Theodoridis, Y., & Tsotras, V. J. (2009). Spatial indexing techniques. In *Encyclopedia of Database Systems* (pp. 2702–2707). Springer.

Mendonca, P., & Goodwin, S. (2015). C-Theta*: Cluster based path-planning on grids. In *Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence* (pp. 605–608). IEEE.

Mitchell, J. S. (1988). An algorithmic approach to some problems in terrain navigation. *Artificial Intelligence*, *37*(1-3), 171–201.

Mitchell, J. S., & Keirsey, D. M. (1984). Planning strategic paths through variable terrain data. In *Proceedings of SPIE 0485, Applications of Artificial Intelligence I* (Vol. 485, pp. 172–179).

Mitchell, J. S., & Papadimitriou, C. H. (1991). The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, *38*(1), 18–73.

Mononen, M. (2010). *Simple stupid funnel algorithm.* Retrieved 2017-04-26, from `http://digestingduck.blogspot.fi/2010/03/simple-stupid-funnel-algorithm.html`

Nash, A., Daniel, K., Koenig, S., & Felner, A. (2007). Theta*: Any-angle path planning on grids. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 1177–1183). AAAI Press.

Navicom. (2017). *Ajoneuvoseuranta.* Retrieved 2017-01-26, from `https://www.navicom.fi/seuranta`

Nievergelt, J., Hinterberger, H., & Sevcik, K. C. (1984). The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, *9*(1), 38–71.

Open Geospatial Consortium. (2017a). *Geography Markup Language.* Retrieved 2017-05-10, from http://www.opengeospatial.org/standards/gml

Open Geospatial Consortium. (2017b). *Welcome to the OGC.* Retrieved 2017-03-01, from http://www.opengeospatial.org/

Open Source Geospatial Foundation. (2017a). *GDAL - Geospatial Data Abstraction Library.* Retrieved 2017-05-09, from http://gdal.org/

Open Source Geospatial Foundation. (2017b). *MapInfo TAB and MIF/MID.* Retrieved 2017-05-10, from http://www.gdal.org/drv_mitab.html

Paikkatietokeskus. (2017). *Korkeusmallit.* Retrieved 2017-05-12, from http://www.fgi.fi/fgi/fi/teemat/korkeusmallit

Panigrahi, N. (2014). *Computing in geographic information systems.* CRC Press.

Patel, A. (2017a). *Heuristics.* Retrieved 2017-04-17, from http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Patel, A. (2017b). *Hexagonal Grids.* Retrieved 2017-04-22, from http://www.redblobgames.com/grids/hexagons/

Patel, A. (2017c). *Navigation Meshes.* Retrieved 2017-04-26, from http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html#navigation-meshes

Pitney Bowes. (2017). *MapInfo Pro.* Retrieved 2017-02-28, from http://www.pitneybowes.com/us/location-intelligence/geographic-information-systems/mapinfo-pro.html

PyGame developers. (2017). *PyGame.* Retrieved 2017-05-09, from http://pygame.org/

QGIS development team. (2017). *Features.* Retrieved 2017-02-28, from https://docs.qgis.org/2.8/en/docs/user_manual/preamble/features.html

Rees, W. (2004). Least-cost paths in mountainous terrain. *Computers & Geosciences*, *30*(3), 203–209.

Richbourg, R., Rowe, N., Zyda, M., & McGhee, R. (1987). Solving global two-dimensional routing problems using Snell's law and A* search. In *Proceedings of*

The 1987 IEEE International Conference on Robotics and Automation (Vol. 4, pp. 1631–1636).

Rogers, E. M. (2003). *Diffusion of innovations* (5th ed.). Free Press.

Ruohonen, K. (2013). *Graph theory.* Tampere University of Technology. Retrieved 2017-4-12, from `http://math.tut.fi/~ruohonen/GT_English.pdf`

Samet, H. (1995). Spatial data structures. In *Modern Database Systems: The Object Model, Interoperability, and Beyond* (pp. 361–385).

Sanastokeskus TSK. (2014). *Geoinformatiikan sanasto* (3rd ed.). Retrieved 2017-5-15, from `http://www.tsk.fi/tiedostot/pdf/GeoinformatiikanSanasto.pdf`

Schneider, M. (2009). Spatial data types. In *Encyclopedia of Database Systems* (pp. 2698–2702). Springer.

Steiniger, S., & Hunter, A. J. S. (2013). The 2012 free and open source GIS software map - A guide to facilitate research, development, and adoption. *Computers, Environment and Urban Systems*, *39*, 136–150.

Tampereen seudun joukkoliikenne. (2017a). *Lissu Liikenteenseuranta.* Retrieved 2017-5-15, from `http://lissu.tampere.fi/`

Tampereen seudun joukkoliikenne. (2017b). *Repa Reittiopas.* Retrieved 2017-5-15, from `http://reittiopas.tampere.fi/`

The Open Source Geospatial Foundation. (2017a). *About FOSS4G Boston 2017.* Retrieved 2017-02-28, from `http://2017.foss4g.org/about/`

The Open Source Geospatial Foundation. (2017b). *About the Open Source Geospatial Foundation.* Retrieved 2017-02-28, from `http://www.osgeo.org/content/foundation/about.html`

Tilastokeskus. (2017a). *Statistics eXplorer.* Retrieved 2017-02-1, from `http://pxweb2.stat.fi/explorer/vaestolaskenta/Vaestolaskenta/index.html`

Tilastokeskus. (2017b). *Väestölaskentatietoja 1987-2010.* Retrieved 2017-02-1, from `http://pxweb2.stat.fi/Database/Explorer/Vaestolaskenta/Vaestolaskenta_fi.asp`

Under Armour. (2015). *Under Armour Acquires Endomondo and MyFitnessPal to Establish the World's Largest Digital Health and Fitness Community* [Press release]. Retrieved from http://www.uabiz.com/releasedetail.cfm?ReleaseID=894685

Van Bemmelen, J., Quak, W., Van Hekken, M., & Van Oosterom, P. (1993). Vector vs. raster-based algorithms for cross country movement planning. In *Proceedings of the 11th International Symposium on Computer-Assisted Cartography* (pp. 304–317). ASPRS.

Van Kreveld, M., Nievergelt, J., Roos, T., & Widmayer, P. (1997). *Algorithmic Foundations of Geographic Information Systems.* Springer.

Van Toll, W., Triesscheijn, R., Kallmann, M., Oliva, R., Pelechano, N., Pettré, J., & Geraerts, R. (2016). A comparative study of navigation meshes. In *Proceedings of the 9th International Conference on Motion in Games* (pp. 91–100). ACM.

Vossepoel, A., & Smeulders, A. (1982). Vector code probability and metrication error in the representation of straight lines of finite length. *Computer Graphics and Image Processing*, *20*(4), 347 - 364.

Wikimedia Commons. (2007). *Vector-raster-vector-conversie.* Retrieved 2017-5-15, from https://commons.wikimedia.org/wiki/File:Vector-raster-vector-conversie.PNG

Wikimedia Commons. (2008). *Tissot indicatrix world map Mercator proj.svg.* Retrieved 2017-5-15, from https://commons.wikimedia.org/wiki/File:Tissot_indicatrix_world_map_Mercator_proj.svg

Wikimedia Commons. (2009). *Category: Images of map projections.* Retrieved 2017-5-15, from https://commons.wikimedia.org/wiki/Category:Images_of_map_projections

Wikimedia Commons. (2013). *Division of the Earth into Gauss-Krueger zones - Globe.svg.* Retrieved 2017-5-15, from https://commons.wikimedia.org/wiki/File:Division_of_the_Earth_into_Gauss-Krueger_zones_-_Globe.svg

Wilson, J. P. (2012). Digital terrain modeling. *Geomorphology*, *137*(1), 107–121.

Xu, X. (2017). *PathFinding.js.* Retrieved 2017-5-16, from https://qiao.github.io/PathFinding.js/visual/

Yap, P. (2002a). Grid-based path-finding. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence* (pp. 44–55). Springer-Verlag.

Yap, P. (2002b). New ideas in pathfinding. In *Proceedings of AAAI Spring Symposium: Artificial Intelligence and Interactive Entertainment* (pp. 95–97). AAAI Press.